EPISODE 648

[INTRODUCTION]

[0:00:00.3] JM: Klarna is a payments company headquartered in Sweden. Since being established in 2005, Klarna has grown to handling $21 billion in online sales as of last year. Roughly 40% of all ecommerce sales in Sweden go through Klarna. Klarna's original differentiator was that it allowed users to checkout of ecommerce stores without entering in credit card information. Instead the user enters an email address and registers with Klarna. This allows Klarna to assume the risk of the transaction in place of the credit card company.

Klarna's clever payment method became very popular, and 13 years later Klarna is actually a bank with the variety of financial services and payment methods. *Marcus* Granström is a director of engineering at Klarna. His work ranges from product development, to systems architecture, to management, and this cross-functionality of the role has some similarity to Raylene Yung from Stripe, who is also an engineering director at a payments company and was on the show yesterday.

Marcus walked me through the life of a payment hitting Klarna servers, and this was a nice starting point for a conversation about Klarna's infrastructure, their product and their engineering practices. I enjoyed the show and I hope you enjoy it as well.

Before I get started I want to mention that we're hiring a creative operations lead for Software Engineering Daily. If you're interested in that role, check it out at softwareengineeringdaily.com/jobs. This is a great job for someone who just graduated a coding boot camp or someone who has a background in the arts. If you want to be creative and you want to learn more about engineering, check it out at softwareengineeringdaily.com/jobs.

[SPONSOR MESSAGE]

[0:01:55.8] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes. You can quickly provision clusters to

be up and running in no time while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked into any one vendor or resource. You can continue to work with the tools that you already know, such as Helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications offline. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes. That e-book is available at aka.ms/sedaily.

[INTERVIEW]

[0:03:31.6] JM: Marcus Granström, you are the director of engineering at Klarna. Welcome to Software Engineering Daily.

[0:03:36.5] MG: Thank you. Thanks for having me.

[0:03:38.8] JM: I want to start with a high-level description of what Klarna is. I think there're probably some people listening who are unfamiliar with it. So the core Klarna product is a payment solution. It's a checkout solution, for example, for shoppers on the internet and it allows them to do things like pay later, or slice up a payment. Can you describe what the core Klarna product does?

[0:04:05.5] MG: Sure, absolutely. So if we start from the merchant's point of view, we basically have a two set of products that we offer to our merchant side, that we offer a full-blown checkout where we do credit card direct debit. It's pretty much any payment options you can think of. Then of course, our own credit products, which is Pay Later and Slice It, which is when we offer

so you can pay off the delivery or you can slice up your payment over in installments or base accounts or different – Depends on market debit.

We also then offer what we call the Klarna Payments, where we sell individual payment methods, so which we usually do for larger enterprises where they have their own credit integration for instance and just want credit options or they just want direct debit. So we can either do the full thing or smaller packages of it.

[0:04:55.2] JM: There was this period of time in maybe the last 2 to 8 years where there was this revolution of online payment solutions and fintech companies. Klarna was certainly in that wave of companies. Why was there such a revolution in the way that online payments occurred and why hadn't that problem been solved in the first 10 or 15 years of the internet?

[0:05:22.5] MG: I think that's when ecommerce really took off and when people who are perhaps not as tech savvy started using the internet more in my mind at least. So if you talk about the history of Klarna, it basically started with providing a way to pay with invoice in Sweden when we bought online, because people were afraid of handing out their credit card and stuff like that. That's how Klarna got started. Then for us then it grew from there out to new markets, out to new payments methods and stuff like that. I think people want a different solution than their sort of regular credit card, which is not as strong in Europe as it is in France, as in the U.S. where credit card has such a large grip of market.

[0:06:06.3] JM: So Klarna is a way for paying without a credit card. Is that what you're saying?

[0:06:09.8] MG: That is what I'm saying. So we basically – For instance, if we offer Pay Later, which is something that's very successful in, for instance, fashion retail, where we buy clothes that you want to really try out before we actually commit and pay for them. You can pay with our Pay Later product. You order your stuff for instance in ASOS in the U.K. and U.S and then get your stuff shipped home. Try it out, fits great, and then you get a payment instruction from us how to settle your fee towards Klarna, because we have ordered to pay the merchants, sort of guaranteeing their income. Then you would have a debt towards Klarna.

[0:06:46.9] JM: Right. Here we start to see some of the complexities of the payment system from Klarna's point of view. So obviously you simplify it for the user, because they don't have to pay immediately. They get their clothes. They can try it on. But from Klarna's point of view, then that seats you with all kinds of interesting problems. Like if you've now got a debt on your books, the consumer owes you money. You have to have some information about the consumer in order to evaluate whether the consumer is credit worthy in order – Should you even give the consumer the right to buy these shoes. You've got to figure out how long should we give the consumer to pay us back. What's the interest rate that we should charge? Et cetera. So there's all kinds of technical complexities that stem from that simple user experience and then that gives rise to, I would assume, lots of engineering problems and challenges.

[0:07:40.4] MG: Absolutely. I mean, that's one of the things that makes it hard, I guess, to offer credit online, because you can't really look at the person and look at their I.D. and know who they are. So we have different ways of both identifying, authenticating users to be able to extend them credit. Then we of course use different means of obtaining data around for the consumers so we can go to external lookup agencies if needed or we have information about you already, because you're already a customer.

[0:08:14.1] JM: Yeah. This is one of the interesting privacy tradeoff things where I think a lot of the discussion around privacy these days is like I don't want to give up my data, because I'm going to be – A health insurance company is going to raise my rates, or I'm going to get advertisements that I don't want to give my information to some company that's going to advertise strange things to me.

I think people often overlook the benefits of giving away data. Like you might get a better credit rate, or if you get your health data way more wantonly, then maybe your healthcare provider can find solutions about you more aggressively. I don't know. The credit opportunity is for giving away your data. I think that's a contemporary example of where it actually you could benefit a lot from giving away more data. At least the kinds of – I think the kinds of people that are listening to this podcast probably will generally benefit from giving away more data.

[0:09:07.2] MG: Yeah, but at the same time it's a super serious and advanced topic where we spend sort of the data we have about consumers is sort of our number one priority to protecting

in all means. We don't – Because that's sort of our secret with the consumer. We really never share it with anyone else and we try to sort of phase it out when we don't need it anymore and stuff like that. But like you say, it's what makes our business possible. If we don't have any information about you, we will not be able to extend your credit basically, because then you would be a blank person.

[0:09:48.6] JM: I think I read something, it's like 40% or 60% of payments in Europe. What's that mind-blowing stat, that like how many payments or what percentage of payments go through Klarna?

[0:10:00.5] MG: Yeah. I think what you're referring to is 40% of the payments in Sweden.

[0:10:05.7] JM: In Sweden, right.

[0:10:06.2] MG: Then online payments goes through Klarna. I think with us being 100% insurer. I think we're around 10% overall in Europe. We do quite significant volume sort of.

[0:10:16.4] JM: So that's enough of a sample size where you could develop so much informa – And you've been going for like 8 years or something like that, 9 nine years maybe, where you've got a good enough sample size where just from Klarna data alone, you can probably evaluate who is credit worthy with a lot of accuracy. How much at this point do you need to go out to sort of third-party data solutions versus just kind of using the data you have in-house?

[0:10:43.3] MG: No. We use a lot of data in-house. We go out sometimes to a credit agency mostly to verify. So if you claim to be someone and you claim this is your phone number for instance, we can go out to lookup agencies and sort of verify that to be able to verify that you actually are a human, not some fraudster trying to use someone else's persona.

In general, we have a lot of data. We of course do a lot of data on where you live and demographics and such stuff as well. But it's not something that we sort of advertise or talk about how exactly we do on credit policies.

[0:11:22.4] JM: Sure. Sure. Yeah. I mean, we don't need to go into that. But what I think is interesting is – So you get this model that's working really well where people can basically pay with an email address or a phone number or some kind of identifying information like that that is more frictionless than a credit card.

[0:11:41.1] MG: Exactly, and we want to – I mean, we really want to try to remove all the friction we can. That's basically our goal to create a smooth experience for when you do payments. So the less friction, the better is where we go. So if it's possible only required email to basically take you on as a customer. That would be the ultimate thing. We do that in a lot of cases, especially if you're a returning customer. Then we know you're usually – Your browser, and then it's so much easier.

[0:12:09.9] JM: I want to spend a little more time laying out the business model and then we can get into some engineering problems. I think we've laid out the fact that there are people who can pay now and then they get basically a debt to Klarna. The merchant gets paid immediately. So here we're already kind of laying out this fact that you have all these debts on the books that you can kind of figure out how to play and you've got a bunch of information about users that you have to aggregate. So you've got kind of a data engineering problem that we can start to think about.

So from the business perspective, today Klarna is actually a bank. So you're classified as a bank. Can you talk about the difference between a bank and a fintech company and maybe just describe the journey from being this fintech company where you have this really well-defined payment solution that was working really well. Then you're like, "Oh! Let's just become a bank." What that classification even mean? I think of a bank as kind of like a changing and weird old world definition. But I'd love to hear your thoughts on that semantic difference.

[0:13:09.9] MG: Yes. I mean, for us, so far it has meant fairly little difference. We're currently exploring the opportunities that we have us being a bank and we're starting to offer some other products. The journey we've been on has been basically that we started looking a lot how we can make consumers happier in general, and then a lot of those things come with providing products that we would need a banking license for.

It's a slow and long process acquiring a banking license, but so far it's been very beneficial. We're just in the process of launching our own credit card. We just launched it to a portion of our consumers in Sweden. So we started to score some opportunities there. We'll roll it out across Europe in the coming months.

[0:13:55.9] JM: That's like a Visa, or a Mastercard, or something?

[0:13:58.0] MG: That is a Visa card, [inaudible 0:13:59.0] who is one of our owners. They took 1% stake in Klarna about a year, a year and a half ago.

[0:14:05.9] JM: Cool. This is something I should know by now, but why is it that Visa and Mastercard or so dominate in terms of the credit card space? Why isn't there more credit card companies, or do you just not even need to invent a credit card? This is something like AWS where it's like it doesn't make sense for us to stand up our own cloud provider. We should just leverage it, leverage the existing infrastructure, because really we would gain nothing by setting up our own credit card network that Visa has already done.

[0:14:35.5] MG: Yeah, I think the investment required to do Visa and Mastercard sort of level penetration on a market, it's just too big for pretty much anyone these days. Just think about how would you get your credit card into the machines. All those millions and millions of merchants. Who would trust it? I think that ship has sailed, basically.

[0:14:59.3] JM: Right. Okay. So the banking license, maybe it takes some time to get, but it doesn't really affect your operations or your engineering. It really just affects kind of the products that you can offer. Can you offer mortgages, for example? You can't do that until you're a bank.

[0:15:16.3] MG: No, we definitely could. Theoretically we could offer a mortgage. We can also offer higher amounts. So when it comes to a virtual consumer, one thing that we've been doing is reducing the credit cards. One thing we done towards merchants or businesses is more that we started issuing credit and loans to them basically to help them grow faster. Getting a loan at a conventional bank today for a small medium business is a pain. The interests are high. The process if really old-fashioned and not sort of suitable to online companies.

So what we started doing in a small scale so far, which we'll start also pushing out is basically providing loans initially to our merchants, but also eventually to other companies where we can leverage the knowledge we have about giving credit and risk analysis in a short amount of time to basically apply online and instantly receive your money basically for businesses as well.

[0:16:12.2] JM: Yeah. I've really thought seriously about taking on debt for my company, but whenever I log in to QuickBooks or whenever I like log in to PayPal, it's like, "Hey, do you want to borrow $100,000 at this really low interest rate?" I'm like, "That's actually a really low interest rate," and I'm sometimes very tempted to just like, "Oh, I'll just take the 100K. Why not?"

[0:16:34.3] MG: Yeah. No, I think we spotted a problem for our merchants. Basically, the biggest problem they have when they're growing is that they don't have enough cash at hand to buy more stuff to sell, basically.

[0:16:46.1] JM: Certainly for apparel companies. I can imagine that being a much more significant problem than a podcast company.

[0:16:51.5] MG: Yeah, exactly. I don't think they have stuff much in common. That's where we basically can help them to grow faster. Of course, in return, we will grow with it.

[SPONSOR MESSAGE]

[0:17:08.3] JM: DigitalOcean is a reliable, easy to use cloud provider. I've used DigitalOcean for years whenever I want to get an application off the ground quickly, and I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU optimized droplets, perfect for highly active frontend servers or CICD workloads, and running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances

have gone down too. You can check out all their new deals by going to do.co/sedaily, and as a bonus to our listeners, you will get $100 in credit to use over 60 days. That's a lot of money to experiment with. You can make a hundred dollars go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure, and that includes load balancers, object storage. DigitalOcean Spaces is a great new product that provides object storage, of course, computation.

Get your free $100 credit at do.co/sedaily, and thanks to DigitalOcean for being a sponsor. The cofounder of DigitalOcean, Moisey Uretsky, was one of the first people I interviewed, and his interview was really inspirational for me. So I've always thought of DigitalOcean as a pretty inspirational company. So thank you, DigitalOcean.

[INTERVIEW CONTINUED]

[0:19:14.9] JM: Okay. So when I go to an apparel company and I purchase something through Klarna, what's happening on the backend when I'm making that purchase? What's going on in Klarna's infrastructure?

[0:19:26.4] MG: So basically what happens is that – Let's say it's a checkout solution. So we own the JavaScript that is loaded, basically the whole checkout that is loaded on the merchant side though is owned by Klarna, because we, for regularly reasons, need to provide the terms and conditions and we need to prove that you actually have the chance to view and accept those terms and conditions. So we can't put full in the merchant's sense.

So the JavaScript is loaded to us. When it comes to the backend, we sort of first identify you, who you are as a person and then we decide what payment method to offer you. So let's say we find you very credit worthy, then we will offer you all the different credit options, Pay Later, or Slice It. If we find that you perhaps is not the most credit worthy person because you just took that $100,000 and haven't really paid it back, then we would probably only offer you to pay with direct payment, and that's just like bank transfer, credit card, or something like that.

[0:20:17.8] JM: Okay. So let's say I clicked to Pay Later, and so there's then some kind of email gets sent to me. What are the other things that are happening on your backend?

[0:20:26.9] MG: Yeah. Like the Pay Later, then it usually depends on – That it's a pay after deliver, like a 14-day credit. So at that point we just enter that order in our system and then we don't actually do anything for a while until the merchants tells us through their sort of integration, the post-purchase integration tells us, "Hey, we shipped the good." Then we actually issue the debt in our systems and it's also when we send the first sort of payment information letter or email where we tell you, "Hey, your goods are on its way. You should receive them any day. Here's how you settle your invoice or debt to Klarna."

[0:21:10.2] JM: So that payment is waiting in – Like it's in a database entry or a queue or something and then –

[0:21:16.6] MG: Yes. What we internally call our issuing systems basically. So we have several issuing system, depending on country. Then it sits there and basically waits for your payment, which the majority of our consumers do to our app in their phone.

[0:21:31.8] JM: Are there any unique database requirements for that? What are you using? Like Mongo, or DynamoDB or something?

[0:21:41.2] MG: Do it depends actually. All these issuing systems are a bit different, but you can say that all our order management actually lies in Dynamo. So we use AWS for a lot of other stuff. So it relies heavily on DynamoDB. So that's where we store it. When it comes to the actual issuing systems, they run on both PostgreS and also one that works on an Oracle database. So it's a bit different.

[0:22:07.1] JM: Interesting. Is it the same data that's sitting in Dynamo and –

[0:22:10.7] MG: Yes and no. So we basically try to decouple the purchase experience. So the issuing systems aren't really effective when a purchase happens. So a purchase comes in through checkout. This goes through the risk decision, and then it ends up in our order management system, and that's where the purchase or the synchronous purchase flow ends. Then we do event-based to inform the issuing systems only about the consumer and their potential debt, because the merchant's information is managed by the order management

system [inaudible 0:22:43.0]. Also there's a clear decoupling between usually what you would say if you talk about a credit scheme, sort of the acquiring and issuing part. All that is decoupled in our world as well.

[0:22:54.8] JM: Okay, and that event trigger – This is like when the consumer makes the purchase, it gets synchronously written to Dynamo and then it also gets written to some relational database, it sounds like, and then you have an event trigger somewhere that –

[0:23:09.6] MG: Yeah. Basically the Dynamo triggers the event, and then the issuing system then picks that up if it's for their market, and then puts it into their relational database. When the merchant them captures and calls our order management system, there's another event triggered saying, "Hey, this order with the order ID was just shipped." They sort of activated that then, and that in turn then triggers a bunch of different events depending on what payment method they have and stuff. If it's a Pay Later, we send them payment instruction. If it's someone that has a base account with us, we just add it to their monthly invoice. Then they get their payment instructions whenever during the month they set up their payment.

It's more or less completed, decoupled from the order management and from the merchant side. The acquiring is sort of standalone. That's why we can have one sort of acquiring system, but several different issuing systems, because they don't really have a dependency between them.

[0:24:06.1] JM: Can you define those terms; acquiring system and issuing and system?

[0:24:09.7] MG: Sure. If we talk about the traditional credit card scheme, and acquirer would be one that sort of brings in the transaction sort of, and the issuer would be the one that issues the credit card, pretty much your bank, because they will hold the balance for your account. It goes the same for us. For instance, in 2017, we acquired a company called Bill Pay, which is a competitor to ours in Germany. Then they of course have sites that run where you pay with Bill Pay. We then sometimes use Bill Pay as an acquirer, but then have Klarna as the issuer. So we can actually have different acquirers and different issuers in our systems. I don't know if it makes sense the way I tried to explain it.

[0:24:54.6] JM: I think it's probably beyond the scope of this particular episode. But talking to some of these fintech companies has made me just realized that I think I need to go a little deeper on some of these like – Just the banking and credit infrastructure, and that's more of a characteristic of this entire category of fintech type of companies and it's not really in the scope or something that's like differentiating for Klarna. It's like –

[0:25:17.2] MG: I agree, and I mean it's a very niche, it's a domain to be. If you don't work and if you really have no idea or usually no interest in knowing these stuff as well.

[0:25:27.9] JM: But it's curious. Like the banking/credit card Rails, they're kind of strange and niche. Are they changing at all? Is that infrastructure going to change in the future or do you see any reason why it would change?

[0:25:39.1] MG: I honestly don't think so. I think sort of if you talk about the Visa scheme and the Mastercard scheme, they of course do small improvements all the time, but it's such a beast. I think it will stay that way. My guess is that the credit card part of payments will reduce a bit over time. We always have other payments methods as well sort of replaced, I think we see that a lot in India and China where you have different payment options sort of in bricks and mortar stores than we do in Europe and the U.S.

[0:26:11.4] JM: Yeah. So that event triggering, going back to the event triggering. Do you kind of have these different events that can be kicked off throughout your system to just propagate a payment or the different side effects of a payment, you kind of have these different events are going to trigger. You send an email based off of some circumstance or send a message to the merchant's side of things based on some event trigger. How do you do event triggering? What's the general pattern? Are you using Lambda or Kafka? What kinds of stuff are you using?

[0:26:43.8] MG: Good question. We're on a fairly large Kafka cluster that we use for pretty much all the non-synchronous communication we do. Sort of can be internal event sourcing that some teams do or can be events to notify other about what's happening in the ecosystem. That's a general sort of pattern that we use. We try to avoid the sort of tight coupling with the synchronous calls. So we do mostly events, but not necessarily, which is very rarely basically. So only the synchronous purchase which has sort of time requirements on.

[0:27:18.9] JM: Is there like an event listeners somewhere? They're listening for the Kafka topics to be updated?

[0:27:26.3] MG: Absolutely. There's, for instance, the place order event, which is the event that I talked about before when the order management system has sort of accepted the order and everything is done in the purchase flow. They should just order place them. I think it's over 10 different systems that's currently listeners to that event for different reasons. It can be for analysis, it can for the issuing of the debt, like we talked about. It can be for sort of update in the consumer side of their app. So we have a big app in both Google Play and the App Store where you can download that, and that also needs to be sort of – Has its own sort of view of orders, that listens to orders or some stuff like that. So it's up to the teams what to listen on and what to discover.

[0:28:13.8] JM: What have you learned about operating a large Kafka cluster?

[0:28:18.6] MG: That's a good question. This is not something I am expert in. But I think we've fairly quickly realized that you can configure it either for throughput or for latency. You can't at least, to my knowledge, have both at the same time. We're very focused on throughput, because latency we don't do or we shouldn't do latency –

[0:28:40.8] JM: Throughput, like reliability you mean?

[0:28:42.8] MG: Yeah, reliability, and sort of volume. The amount of events that we can push through the cluster before it gets problematic.

[0:28:51.0] JM: Like order consistency as supposed to like low-latency and questionable ordering.

[0:28:55.9] MG: Yeah, exactly, and we deliver one sort of promise. That's definitely something that takes where we are. But like I said, I'm really not into the details there. But that is what I from my limit and knowledge know at least.

[0:29:11.2] JM: No problem. So that alone is an interesting tidbit. I'm sure it's useful. So we kind of talked about the transactionality of just payment going through and how you kind of write to different databases and how that trigger updates different constituents that are interested across your infrastructure, and I'm sure there's a lot more meat to that. But I'd like to talk more generally about the platform engineering for Klarna. So I mean you got data scientists and data engineers and people that are writing different frontends and people that are spinning up experiments. Do you have a standardized – There're obviously releases for all these things too. So do you have like a standardized platform engineering team that defines what cloud providers we use, what languages we use. Do you have any kinds of standardization across the organization?

[0:30:05.0] MG: Yes, we do. We have an architect board, which I'm personally a member of and that consists about 12 people. We meet once a week to discuss everything from those topics sort of programming languages to which databases do we want to invest in. What makes a good event basically? How do you provide enough data in an event in a good way and how – Stuff like that.

So we're fairly standardized when it comes to programming languages, when it comes to databases, cloud providers and stuff like that. We sort of have to be with the amount of sort of engineers we have and what we want to sort of achieve by doing that. It is of course a bit limiting to teams that we have to choose between four different programming languages, but the flexibility that it gives us both with the possibility for engineers to new teams, but also to new services to other teams if we change the mission of one team and then sort of this sort of overweighs that the teams can do whatever they want.

We have that approach previously, where teams were more free in the technology choices and especially the programming language choices, and we had so much pain when we want to move stuff around. So then we basically restrict it to Java, JavaScript, Scala and Erlang as the four languages that we sort of invest. Then we have – Where we also have very –

[0:31:29.8] JM: Sorry. You said Ruby, Scala, Erlang?

[0:31:30.7] MG: No. Sorry. Java, Scala, JavaScript and Erlang. So where we also have in-house sort of expertise on all those languages. So for a team running to a corner case somewhere, there will be someone internally that we can ask, and the same goes with databases require. Committed to PostgreS and PostgreS RDS, and we have some really hardcore PostgreS debuggers if needed internally. That's the only reason we've gone more or less with PostgreS, is that we have these experts in-house and that why would anybody go with MySQL when we have PostgreS experts in-house. That's sort of been the reason behind that.

[0:32:13.5] JM: Scala and Erlang. Tell me about that.

[0:32:15.8] MG: So Klarna is a functional shop from the start. So when Klarna was founded, Klarna has three founders. They're all non-technical. They can't build the solutions. They wanted their own. So they asked for help from their first investor. She had worked at Ericson for a longtime. She knew a bunch of Ericson engineers that now run a consultancy shop. So they basically came in and built the embryo of Klarna in Erlang, and it actually great from there. So for a long while we were an Erlang only shop.

I think more or less everyone – I don't if it's true anymore, but five years ago, everybody that works with Erlang had or worked at Klarna at some point in their career in Europe. Obviously that became a bottleneck when it came to recruitment. So then we started sort of looking outside of the Erlang bubble. Scala had in recent year become a good sort of functional language that we also use, then we initially went to Java, which is just a big work course in a lot of companies.

[0:33:25.1] JM: Do you find that when you're hiring people, that Erlang and Scala side of your infrastructure is like something that can get people really excited about working at Klarna, or it can make them like really anxious about potentially working there?

[0:33:40.3] MG: No. I think it's both. We definitely have people who have come to work for Klarna because of our Erlang stack, and I don't think we – And those are the people we hire to work there. We don't push a linear JavaScript engineer into their Erlang community straight away. We ease them into it a bit. But it's a definitely [inaudible 0:34:00.2] for some of the engineers that really like and appreciate functional program.

[SPONSOR MESSAGE]

[0:34:13.2] JM: Nobody becomes a developer to solve bugs. We like to develop software because we like to be creative. We like to build new things, but debugging is an unavoidable part of most developers' lives. So you might as well do it as best as you can. You might as well debug as efficiently as you can. Now you can drastically cut the time that it takes you to debug.

Rookout rapid production debugging allows developers to track down issues in production without any additional coding. Any redeployment, you don't have to restart your app. Classic debuggers can be difficult to set up, and with the debugger, you often aren't testing the code in a production environment. You're testing it on your own machine or in a staging server.

Rookout lets you debug issues as they are occurring in production. Rookout is modern debugging. You can insert Rookout non-breaking breakpoints to immediately collect any piece of data from your live code and pipeline it anywhere. Even if you never thought about it before or you didn't create instrumentation to collect it, you can insert these nonbreaking breakpoints on the fly.

Go to rook out.com/sedaily to start a free trial and see how Rookout works. See how much debugging time you can save with this futuristic debugging tool. Rookout integrates with modern tools like Slack, Datadog, Sentry and New Relic.

Try the debugger of the future, try Rookout at @rookout.com/sedaily. That's R-O-O-K-O-U-T.com/sedaily. Thanks to Rookout for being a new sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:36:18.1] JM: My first job out of school, the company had a lot of infrastructure in Erlang, and before I was going to start at the company, I spent a couple of weeks trying to learn Erlang and I was having so much trouble and I was just like really worried about going to work there and just like, "Oh, no! I don't know Erlang. I can't figure it out." Luckily they of course had plenty of other programming languages that they were using, but it kind of scared me.

[0:36:41.6] MG: Yeah, it's a steep learning curve in the beginning, but I'm now an Erlang guy. I came in on the Java and Scale side, but it's a steep learning curve. But we have people internally who have switched from both the Java and JavaScript to Erlang and, yeah, they seem to like it.

[0:36:59.3] JM: What's your cloud provider stance?

[0:37:02.2] MG: So at the moment we run more or less exclusively on AWS. We are more or less 100% a cloud based company. We still have some legacy and running on premise, but we're moving everything into the cloud. We're experimenting a bit with running stuff on Google cloud sort of in parallel, but at the moment we're currently heavily invested into AWS.

[0:37:26.0] JM: The most common service that I hear people using out of Google cloud is BigQuery. So it sounds like you have some interesting data engineering problems. So has BigQuery been a potential Google cloud experiment?

[0:37:40.7] MG: I would say no. I would say. I absolutely did it. It's a great product and I think we could definitely leverage it. But I think the way we've been experimenting is more that we will have – Not a backup, but sort of being able to route traffic between different cloud providers.

So we are not completely locked into AWS and completely locked into Amazon. So Amazon is both a partner and a competitor, and some of our merchant has sometimes sort of expressed a disapproval of us running in AWS, because it's definitely their biggest competitor.

[0:38:15.8] JM: Oh yeah, I heard about that kind of concern, like a hardware store doesn't want – Put their online store on AWS, because they're afraid that Amazon is going to like look into their information or something like that, which – I don't know. Whatever. That's their decision. Maybe it's not paranoid.

[0:38:34.3] MG: No. I don't think that is a problem, but it's a little bit – Maybe it's wrong to use the phrase feeding the beast, but somewhat paying your competitor, although it's a completely different mark. I don't think they will ever do something that would sort of impact the [inaudible

0:38:50.0] making a bigger and more successful. It's very few merchant that have a [inaudible 0:38:54.6].

[0:38:55.2] JM: Sure.

[0:38:56.6] MG: It's been mentioned.

[0:38:57.9] JM: Well, and it sounds like there's equal concern for the standpoint of – I mean, if AWS is a single point of failure, that's not great. You'd like to have some kind of failover if there's some sort of weird bug that propagates their Amazon's infrastructure. It would be great to have a failover to Google cloud, right?

[0:39:16.4] MG: Absolutely. I mean, it's a very technical hard problem to solve if we do routing outside the both cloud providers. Definitely, it's something we would like to achieve and it's something we've been sort of experimenting with. Then it goes for the same saying goes about routing traffic between different regions, and AWS also is something that we spend some efforts in trying.

[0:39:41.5] JM: So far as platform engineering and deployment of your infrastructure, you got started when, I guess, the state of the art was your deploying to EC2 instance or, well, it sounds like you have some on prem stuff, but then eventually EC2. Then today is it containers, or Kubernetes, or what are you doing there?

[0:40:01.2] MG: We leverage Kubernetes for our sort of tooling. So like big service and all that, but we have decided not to run it in production. We feel that it's too many moving parts. We would probably do it in a heartbeat if we were streaming video or something like that. But when we're managing transactions and people's money, it matters. Perhaps it doesn't matter if you miss 20 seconds of a movie, but if you miss 20 seconds of a transaction, it's a big issue.

We decided not to go there yet. We're looking at sort of the maturity and when we feel that it's easy enough for us to manage it, then we might take the step. But at the moment we run on EC2. We still run Docker containers, yes, it because it provides so much help in running locally in the building and packaging.

[0:40:54.0] JM: What aspect of the Kubernetes solution do you feel would increase the – It sounds like you're kind of worried about like increase in Byzantine failures and losing some transaction times, and maybe due to like Kubernetes networking or something like that.

[0:41:08.3] MG: Yeah, exactly. So networking discover another thing, especially for us as a regular entity. It's sort of access management that you need to put on top of Kubernetes, that at least when I was part of looking into it, didn't sort of come up to the box. So how you – Because how do you know who removed pods and stuff like that? How do you get to trade [inaudible 0:41:31.7] of that stuff? So that is not something that comes out of the box. It needs some layers that you need to add on top. I'm sure there's solutions out there for that already, but it doesn't – For instance, if we would run in ECS, elastic container service, that that stuff works out of the box with IM permissions in AWS.

[0:41:51.3] JM: An auditability.

[0:41:52.3] MG: Yes. And I'm sure if we run on Google cloud, that would also work out of the box. But since we don't, we're probably looking at another solution at the moment.

[0:42:03.9] JM: Are there any advantages? When you look at Kubernetes versus the infrastructure you have today, obviously Kubernetes is kind of like the trendy thing to do. But are there any material advantages when you look at it and you're like, "Oh, it would be really cool if we're on Kubernetes because of X."

[0:42:20.2] MG: I mean, the really integral part is the resource utilization. We're fairly poor on the resource utilization. We have a lot of idle time on our service. So that would be obviously an improvement and what we eventually would like to get out on orchestration settled like Kubernetes or whatever it is. But it's not something that we're losing sleep over either. I mean, if we were not a profitable startup, then Kubernetes would probably be a great option for us to sort of limit costs and stuff like that.

[0:42:53.4] JM: Okay. So getting to the data engineering stuff, so I talked to some companies that they put their whole kind of data engineering stack in PostgreS and there's both pros and

cons to doing that, but I find it an interesting approach, or where they put at least as much data engineering volume into PostgreS as possible, and this sounds like we got a pretty good PostgreS team. What's the data engineering infrastructure like? What kinds of systems are you using, like Spark, or Red Shift, or what kinds of stuff are you using?

[0:43:28.3] MG: This is not either my expertise area, but I know a bit.

[0:43:31.0] JM: That's okay.

[0:43:31.5] MG: We try to leverage a lot of AWS resource and services that they provide. So Red Shift data pipeline and all that, and we also run our own Hadoop cluster sort of for batch processing on the side. But we're moving more and more on what we used to have on premise to AWS, and the data stuff is sort of the lost frontier there. So we're definitely heavy users of Red Shift and the different data processing tools in AWS at the moment.

[0:44:02.5] JM: So we did a show recently with Uber about their data platform, and Uber has kind of an interesting problem where in some ways the data is well-formed, because when somebody takes a ride, okay, you get a driver, you've got a rider, you've got the time that they spent. You got the amount that they pay, et cetera. So it's like a very schematized.

But in other cases, it's less schematized, because your selection of options in Thailand is probably different than your selection of options in Sweden. Maybe in Thailand you have the option of riding like a motorbike, and a motorbike only seats one person and it's cheaper. So you have some schema that is well-defined, and you have some scheme that is less well defined.

They're in this interesting data engineering situation where, because in data engineering, in some situations you want your data in some JSON like object with less schema that's defined upfront. But in other situations, you want it in like a parquet file so you can do rapid aggregations and have it in this well-formed columnar shape. Do you have any interesting decisions around that kind of schema? Because you're offered in different international zones, so I don't know if the data schema varies by the international zone. Is that an interesting problem at all for you?

[0:45:24.4] MG: Not that I'm aware of. That data definitely differs between countries and stuff like that, but it's still pretty well-defined beforehand. So I think we can structure most to our data fairly well, I would say. Like I said, like I mentioned before, data is the part where I get my nose out.

[0:45:42.7] JM: Right. Okay, fair enough. So what are you mostly focused on day-to-day? What are the kinds of engineering problems? Is it mostly hiring problems and setting KPIs for people?

[0:45:53.8] MG: Yeah, so I work and run the domain that's called merchant's services. My focus is mainly on everything we do towards our merchants before the purchase and after the purchase. A bit in the purchase flow, but mainly the other stuff. So it's around how we onboard merchants in a good way. How we do successful underwrite the merchants, because we do underwriting for our merchant to decide if they will have a payment delay, because they sell shoes and they have 40% return. So we don't end up where we have an exposure, a large exposure to a shoe sale merchant for instance and stuff like that.

We also have technical problems like order management and everything that comes with that and how we do merchant lending and the financial reporting card, which is a sort of a beast on its own sometimes. How do you inform the merchants about all their sales without being able to give any details about the consumer [inaudible 0:46:49.8].

[0:46:51.3] JM: Well, I wish I would have asked you this question earlier, because I know we're almost out of time and I gave a bunch of interesting things that I shouldn't have asked you about.

[0:46:59.5] MG: Pandora's box.

[0:47:01.0] JM: You sure did. It's cool, because it sounds like you've got – You spend a lot of time thinking about like product level things perhaps as much as the engineering solutions behind them.

[0:47:09.9] MG: And that's something that when we do another part, which I spend a lot of time, is hiring. So that's one thing we are going through this enormous growth phase at the moment. So I spend personally a lot of time hiring, but I really value in engineers is when I see engineers that cares about the product, because the teams that we organize them, they will own basically any product, or a part of your product, and I want them to care about that, because they make so many decisions on a day-to-day basis that will impact that product. Then they really need to understand why do we have a three-step onboarding, for instance. How does that affect the product and the merchant [inaudible 0:47:53.1] in the end. Stuff like that. It's what I really – Besides all these technical skills and stuff, other culture fit as well. I look a lot about in engineers when I interview them.

[0:48:04.5] JM: Cool. Maybe if there are some other engineering at Klarna that's interested in coming on the show, or if you want to come back sometime in the future we can talk about product engineering, because I think that's the kind of that structure where you have the engineers caring a lot about the product. I think that's a little unique.

I mean, most organizations I talk to, they have the PM layer that sort of takes that off the engineer's hands. I don't know if you have a PM layer.

[0:48:26.6] MG: Yeah, yeah, absolutely. We have PMs in sort of my domain and it's run together with my product counterpart, but really see them as counterpart. It's a give and take and I won't [inaudible 0:48:37.3] for the technical sort of debt or whatever technical roadmap. But I also want them to be fully engaged and fully understand what the product side of things we're building and be able to defend why we decided to build this part of our inside product. [inaudible 0:48:57.0] have to wait from our product manager. They should be so engaged and involved that they should know that.

I know a lot of other companies where engineers are seen more as orders takers and don't have to think about why we're building our products in one or the other, but since we really try to create these autonomous teams that are so self-going and that it becomes important that engineers know why they're doing stuff.

[0:49:27.2] JM: Yeah. I think that will get selecting for engineers, that the kind of engineers you want, those cross-functional ones, the lateral thinkers, etc. Maybe not always, maybe not the PostgreS debuggers. Maybe those people –

[0:49:39.6] MG: No. I mean, there's always room for people with deep technical skills and stuff like that.

[0:49:44.3] JM: Right, of course.

[0:49:46.5] MG: So if we do generalizations or what we're – How the engineers that we hire.

[0:49:51.6] JM: Totally. Cool. Marcus, this has been great. I really enjoyed talking to you.

[0:49:56.0] MG: Well, thanks. It was enjoyable for my side as well. Thanks again for having me.

[END OF INTERVIEW]

[0:50:02.8] JM: If you are building a product for software engineers or you are hiring software engineers, Software Engineering Daily is accepting sponsorships for 2018. Send me an email, jeff@softwareengineeringdaily.com if you're interested. With 23,000 people listening Monday through Friday and the content being fairly selective for a technical listener, Software Engineering Daily is a great way to reach top engineers.

I know that the listeners of Software Engineering Daily are great engineers because I talk to them all the time. I hear from CTOs, CEOs, directors of engineering who listen to the show regularly. I also hear about many, newer, hungry software engineers who are looking to level up quickly and prove themselves. To find out more about sponsoring the show, you can send me an email or tell your marketing director to send me an email, jeff@softwareengineeringdaily.com.

If you're a listener to the show, thank you so much for supporting it through your audienceship. That is quite enough, but if you're interested in taking your support of the show to the next level,

then look at sponsoring the show through your company. Send me an email at jeff@softwareengineeringdaily.com. Thank you.

[END]