# EPISODE 572

[INTRODUCTION]

**[0:00:00.3] JM:** Keybase is a platform for managing public key infrastructure. Keybase's products simplify the complicated process of associating your identity with a public key. Keybase is the subject of the first half of today's show. Michael Maxim an engineer from Keybase gives an overview for how the technology works and what kinds of applications Keybase unlocks.

The second half of today's show is about Clarifai. Clarifai is an AI platform that provides image recognition APIs as a service. Habib Talavatifard explains how Clarifie's infrastructure processes requests and the opportunities for improving the efficiency of that infrastructure. Last month, we had three Software Engineering Daily meetups in New York, Boston and Los Angeles. At each of these meetups, listeners from the SE Daily community got to meet each other and talk about software, what they're building and what they're excited about.

I was happy to be in attendance at each of these and I'm posting the talks given by our presenters. The audio quality is not perfect on these, but it's not terrible and there's also no ads. Thank you to data dog for being a gracious sponsor for providing this space for our meetup and also for sponsoring the podcast. You can sign up for data dog and get a free t-shirt by going to softwareengineeringdaily.com/datadog.

We'd love to have you as part of our community. We will have more meetups eventually and you can be notified of these meetups by signing up for our newsletter. You can also come to softwaredaily.com and get involved with the discussion of episodes and software projects, and you can check out our open source projects, the mobile apps and our website. You can find all of that at software.daily.com.

With that let's get to these weekend episodes about our meet-up presenters.

[INTERVIEW]

**[0:02:05.9] JM:** Thanks for coming to the meetup. I'm really excited to have Mike Maxim and Habib Talavatifard to speak from Keybase and Clarifai respectively. Before we get started, we're going to hear from Datadog who's the sponsor, the host who has generously provided the food and beverage for everybody. Yeah, Ilan, take it away.

**[0:02:27.8] IR:** Thanks everybody for joining us here at Datadog today. Just a quick show of hands, how many folks are using Datadog at work right now? If you work at Datadog, you don't count, because –I mean, you do count. Yeah, so Datadog is a monitoring platform for all your infrastructure and applications, everything from logs to metric logs, to metrics, to tracing. We'd love to help you monitor your infrastructure.

We also love to have you join our team. We're here in New York, as well as in Boston, in Paris and are pretty remote-friendly and we're hiring for everything from SREs, to software devs, to everything else you may be able to imagine. Our website is full of jobs. If you're here during the breaks and you want to learn a little bit more about opportunities here at Datadog, love to chat with you.

Jeremy Dan, raise your hands. Other Datadog people raise your hands. You can find any one of us and chat with us about opportunities here, everything from again, from hacking on back in analytics systems and with Redshift and Looker and Spark and Kafka and everything else in between, to SRE and stability and everything, and other topics as well.

Another thing I wanted to mention is if you look on your seats, most of you seem to have sat on them. There are little scratch off cards. They are actually worth something, so you might want to grab one. Datadog's running our user conference at the end of the – over the summer, July 11th and 12th is an event called Dash, and each of you have a little scratcher ticket on your seats with an opportunity to win a free pass, or discount and passes to Dash. Be a great opportunity to get hands-on workshops on everything from containerization and observability, to seminars from your peers on how they're scaling their systems and their infrastructure, like some of the topics we're going to hear about today.

Without further ado, I'll hand – I don't know if I'm handing to Jeff or I'm handing off to Mike, but back to you all. Thanks.

**[0:04:09.9] JM:** Thank you, Ilan. I think there are some people here who don't really know what Software Engineering Daily is, which is totally fine. Just to fill you in, if you haven't heard of Software Engineering Daily, it's a podcast about software engineering. It's five days a week and the format is fairly technical software engineering content. If you are a listener to the show then hopefully I've had a chance to say hi and shake your hand and meet you. If not, I'd love to meet up, talk a little bit, or grab 30 minutes after the presentations hopefully, a chat for a little bit. As always you can send me an e-mail, or Slack message or whatever.

Yeah, we'll get the ball rolling. The schedule is on the meetup page if you don't know what the schedule is. We'll start with Mike Maxim who is – just to give him a quick introduction. He's been the CTO and the CEO of OkCupid before he joined Keybase, which was started by the founders of OkCupid and Keybase is a pretty incredible company. I think we're going to be hearing a lot more about Keybase in the future, so I would pay attention closely.

**[0:05:15.4] MM:** Thank you, Jeff and thanks for Datadog for having us, having a Keybase here to give you guys some more information about it. Before I get started, just a quick show of hands, how many people have heard of Keybase, or have used Keybase? All right, some people. How many people know what public key cryptography is? More people. All right, great, good.

We won't be starting from scratch here. Keybase, crypto for everyone. Keybase was started in 2015 and the stated mission is to bring public key crypto tools to everyone. The space has traditionally been very complicated, very hard to use and has never really caught on other than SSL or TLS with larger groups of people. The tools have been traditionally hard to use and hard to really coordinate and get people into.

The goal of Keybase is just to change all that, provide an infrastructure to make public key crypto accessible to people and to provide apps on top of that that allow you to integrate crypto into your day-to-day workflow on your computer, either through chat, file system, or with Git for programmers, that thing.

Tis talk, we're going to talk a little bit about crypto. I know people have indicated know this, but we'll just do a quick refresher and talk about how Keybase fits into that scene, talk about how Keybase can run on devices and then these apps that I was just mentioning that take advantage of all this infrastructure that Keybase provides. These slides are public at that address.

Cool, so why is crypto important? This little slide here shows all the services people use. People use a lot of these cloud services these days, anything from Dropbox, Twitter, Facebook, Slack, Google Drive, all kind of stuff. You're communicating through the cloud and you store your files there.

Traditionally all the stuff is out in the open. Once it hits people servers, Slack almost certainly can read all of your e-mail, or all of your stuff, Dropbox can as well. It's fine for a lot of people, but it'd be nice if there is a way to have these cloud services, but not have to make that sacrifice and personal privacy, or to be attacked.

The traditional crypto setup that these little crypto exercises, we have the actors as being Alice and Bob; these are two very famous Alice and Bob people. Then Malory is traditionally your adversary in these things. The goal of crypto is to be able to do all these fun things that you can do without having to give up your privacy, or be attacked by a third party.

What does Keybase do? Keybase is a set of tools, like I said that solves this Malory problem here. Our goal is to create a set of usable tools that allow you to do all these things that you're used to doing; chat, share files, collaborate and host your content in a secure way. Here on the bottom is, well I don't know if you can really see this that well, but these are the four main things that we have in the Keybase app.

The key aspects of true encrypted communication is you want to be able to – Alice wants to be able to communicate with Bob under the following circumstance. You have an adversary called Malory listening in. Malory can possibly tamper with whatever is there, Malory can impersonate other people in this chat, controls the servers, so all this stuff is possible. Alice wants to be able to communicate with people under these circumstances here. How do we how do we make that happen?

The traditional approach here is through public key cryptography, just a brief refresher on how this is, just for people to know more about this is the Diffie-Hellman formulation of public key crypto. The idea here is that you want to both Alice and Bob publish in some way public key associated with both of them, and that public key comes along with a private key that is known only to Alice and Bob. This is called a key pair, so both Alice and Bob have a public/private key key pair, they publish the public key somewhere.

When they want to talk, they get together and they can do some fancy calculation with both of these public keys and come up with a shared secret that they can then use to communicate with using a normal crypto algorithm. For a lot of these formulations, you can use elliptic curve Diffie-Hellman. If people know what that is, that's what Keybase uses, but there's others like RSA that you may have heard of.

Furthermore, if you really want to get fancy, you can actually not have one public key for yourself, but many.

**[0:09:23.4] PARTICIPANT:** Is the shared secret really a big problem with public keys? Because public keys –

**[0:09:29.1] MM:** Public keys are used to compute a shared secret. It's a combination of the public and private keys of both parties. Alice and Bob can do some calculation. In the elliptic curve calculation, you know a point on a curve and then you can multiply it by the public part of the curve and then both of them can do that. It reveals no information in the other party, because in order to do that, they have to solve some very hard problem. In this case, the discrete log problem to back out the private key.

Once you get that calculation, the shared secret could just be H Mac of whatever you compute from your private key and the public key part of your –

**[0:10:03.8] PARTICIPANT:** Still some just the public key.

**[0:10:05.3] MM:** No, it’s a combination of your private and public key. That's good.

**[0:10:08.5] JM:** We'll do some time for questions at the end. No problem.

**[0:10:11.9] MM:** Moving forward, so here's this little diagram of how this would work. This guy over here says, “Dear Alice, I want to talk to you.” They do this key exchange here using their public and private keys to come up with a shared secret. Once the shared secret is established, then that takes over. Now the reason you do this is because these symmetric key algorithms typically are much faster, it's much easier to communicate with somebody in that way, than they have to do some RSA type thing over and over again.

Okay,, so that's the basics there. This has been around for a long time. This isn't anything that Keybase has done that's new. There's a lot of public key crypto software, other things like PGP, or GPG tools, there's been plugins for your mail program. One feature of a lot of these things that they're very difficult to use. I don’t know if people may have received an e-mail in the past that had some PGP header on it that you're supposed to run through your GPG tools and verify the signature of the person who sent it to you.

I don't know. I mean, this thing is probably completely out of you, so I don’t remember the last time I even received one of these things in e-mail. A big part of that is because it's very difficult to use. I mean, nobody really knows what it is, you'd have to integrate it with your mail program in some way. It always breaks, hard to find people's public keys. It's gotten a little bit better recently. There's a lot more, especially for chat, there's tools like Signal, Whatsapp and others that make this a little bit easier, but they're still not great on the on the dissemination of what your public key is.

That takes us by the next point here, which is there's a big – getting the public key, we’ll refer to it as the identity problem here. How does Bob get Alice's public key? Where do you go? For the web, something like TLS, there's these certificate authorities that sign people's certificates and say this is legit. If you're just out in the world, what do you do?

Well, there's these things called key servers for PGP. Here's an example if you search for Gavin Andresen, for people to know he was used to be a Bitcoin Core developer. You go into MIT’s PGP server, which is one of the canonical ones, you search for Gavin Andresen and you get this list of keys. Some of them are labeled revoked. Nobody knows what this is. It's very difficult to actually use this to send Gavin a message, you can be you can be happy that is actually encrypted just for him.

There's other flawed solutions to distribute the keys, you put on a key server, you can post on your website. Problem with that is how they know it's actually your website. You can put it on all your e-mails, that's a little bit hard, doesn't scale to all your friends. You put it on Twitter, problem there is that now you have to trust Twitter. Twitter can lie about what the actual key is, or you can do something called web of trust. PGP has this notion you could go to a key signing party. I don't know if anybody's ever been one of these things, but the idea is that you show up with a laptop and your public/private key pair and you just sign each other's keys.

The point of that, so that if somebody finds your PGP key on the wild we're like, oh Mike's private key, this this key was signed by all of Mike's friends, so it's got to be Mike. Problem with that is nobody goes to these parties. Yeah, Facebook also has a cool feature, where you can go and post your PGP key to Facebook and people can find it that way. Another problem with this is that Facebook and of course lie to you. Facebook's a pretty trusted entity, probably not going to lie. If you really want to do this right, you have to – of course, maybe not these days, but I don't know.

Again, any centralized server that's just giving you a PGP key to be viewed with suspicion. Now how does Keybase solve this problem? The Keybase solution is basically to take the sum total of all of your social networking identities and sum them up as you. Your Twitter account, your Github account, your whatever, your Facebook account, all of these accounts online for Keybase are basically your identity.

Not only are these services bringing your identity together, this is how we're actually going to distribute your key at all. We're going to do it in a way that any pro let the Keybase client can go and look at these services, go look at how you proved that you own these social media accounts and automatically verify it, so that if anybody tries to change it, or if those servers take it down that well, you can know and so forth.

The Keybase service is a centralized server, but the trust for the validity of the identities in the system is decentralized after the clients. In other words, all the information coming from the server should just be viewed as a hint. Whereas, the Keybase clients are going to – then once they received that hint, will actually go out and verify all parts of your identity.

What does a Keybase proof actually look like? As soon as my slide gets up, I will show it to you. Basically what it's going to come down to is your Keybase client going to say, “I want to prove Twitter.” What you're going to do is you're going to tell the Keybase client what your account name is and the Keybase client is going to give you a bunch of text that says, “Post this in and put a tweet up on Twitter, and I'm going to go look for it.”

This tweet has the forum, “I am blah, blah, blah, blah in Keybase and here's a little string here of stuff that verifies that it's me.” That string of stuff is a digital signature. A signature that can only be created by the owner of the public-private key pair associated with my Keybase account. That’s we proving his Twitter account. We can do more than Twitter. We can do a variety of other sites, but first you can look at what this actually looks like for a user. I guess, this is unseeable here, but this is a cig chain link for this user.

Basically, every user on Keybase is represented by a block of these proofs and each one verifies the other. This one is for Twitter. I'll talk about that in more detail on a on a future slide, but here's a proof on Github, here's a user also claiming a Github account, and then you can also claim websites, you can claim DNS, you can claim Facebook, all stuff. The purpose of this is as you post these proofs, all these proofs are then checkable by people's Keybase software to verify that you are who you say you are online.

Here comes a little slide that shows how Alice and Bob will communicate this. Here's Alice talking a Bob saying, “Bob, I want to send you some super secret conditioning formula for Bob.” Bob is able to verify that it's the true Alice by – on her client will say like, All right, Alice has proved her Twitter account and her Github account. I know Alice by those two identities. I'm going to go out to these services, I'm going to verify it’s Alice. As soon as I do, I'm going to accept this as a legit communication from her.”

In addition to these social proofs and the association with your either PGP keys, or these other device keys, which I'm going to get to in a second, you can also do something like PGP’s web of trust here on Keybase, so you can follow other users as well and it gives you the same benefits that you get in the PGP world, except it's a little bit better, because when you follow somebody you have to verify their proofs, so you vouch that this person – if I follow somebody and I see that they have Github proof X and Twitter proof Y, if I then follow them I've vouched for the validity of those proofs, myself as well.

A follower is more than just an entry in the database. I go in and I actually sign a statement with one of my keys and I say, "I'm coming in here." I'm going to say, "This person is who they say they are and here's a signed statement from me to that effect," and I can broadcast this out to the world. These are the two things we have here. We have the social proofs and these web of trust to follow system on the site that lend power to the identity that you have on Keybase that allows you to get to these public keys that people have published.

Okay, so I've been mentioning the Keybase client a lot. The Keybase client is a standalone application that you download and run in your computer. It doesn't run in a web browser, the risk there with anything running in the web browser is that it could possibly can be changed, you never know what you're going to download from JavaScript. JavaScript could change on the server. Whereas, if you just download this client, it only updates when you want it to. It's all open source so you can go and verify everything in there, yourself if you want, or you can rely on a trusted party to know that this is legit software.

Like I mentioned, it doesn't update underneath you if you don't want it to, so that's great. It also allows it, so that we can store your public private key pair associated with this application on your device. That takes me to the next slide here. In the beginning, Keybase was very much like a PGP, almost like a PGP key server. The idea was is you would – you would upload your PGP public key to Keybase, prove a bunch of identities and it would be very easy for you to then find people's PGP keys and potentially send them some – either an encrypted e-mail, or some other out-of-band thing.

What we realized is that people use many devices these days. You have a phone, you got a computer, you got a laptop, there's a lot of different places to where you're going to need keys. PGP keys are notoriously difficult to move around. I mean, the PGP story really is that you should have one master key, which you stick in a explosion-proof safe or something like that.

You never touch it, and then you delegate a bunch of sub keys to actually do – to actually be the ones you use out in the world.

With Keybase, what we decided to do was instead of having to worry about all that, we felt it was too difficult for users to work with is to instead have keys associated with a single device When you install a Keybase software, you automatically get a NACL public-private key pair installed on your device. That that public-private key pair is signed into your account.

Now in addition to having a PGP public-private key associated with a Keybase account, you can also have these NACL keys, which are these ECDH keys also associated with your online identity. That means when you're running your Keybase software, if you go in, you can sign things, you can encrypt things and they're all signed encrypted using the key that was generated from the software that was installed on your machine.

This is pretty great, because this means this private key never leaves the device. You don't have to worry about as a user transferring anything from computer to computer. As long as you have provisioned the device using some existing key beforehand, that device is ready to go and can access all the stuff that you care about in your Keybase world. All these public keys, they're associated with your devices, are all shared publicly. Anybody can encrypt a message to you with any of them, which is very convenient.

It makes things like our chat program, or our file system application work very well. It allows you to provision a new phone for instance and immediately see all your chats. That's not necessarily true in a lot of crypto applications, just because of how this system works, but with how these devices will sign themselves into your identity. They can also be revoked if you lose it. I keep using this term cig chain a lot. Let me just let me just flesh it out a little bit here, just so we can, be sure we know what we're talking about.

A sigchain is basically like a blockchain of your identity. You can have blockchain the way it works as a basically says here's a bunch of data in a row and I want to make sure that you are absolutely certain that this ordering is correct, nobody’s subtracting any blocks out of here, nobody inserted anything that they should. The way they do this is they're able to sign parts of the previous blocks and say this one is legit, because I can see all the previous ones before it and they all have the proper signatures going back in time.

That's basically how these sigchains work as well. Every time you change your identity on Keybase, it is signed into the chain, which represents your identity. If you prove Twitter for instance, that's going to be a link in your sigchain. If you then add a devise, if you add a phone that's the next link and that link will refer to the other one, so that a client can verify that they're all lined up properly.

When the Keybase client loads a user, it will load this entire sigchain and play it back to make sure that the server gave it accurate information. This goes back to what we we're talking about earlier where the server is really just giving hints as to what's happening and the clients are actually going out and verifying that the identities are getting – are correct using these remote proofs, as well as the validity of the sigchain itself.

I have another example here from my brother Jeff, his own sigchain. I use this because it's a particularly good one, it's an interesting one to look at. You can't really see the text here, but the legend is viewable. Basically any of these pink box is a device, green was a PGP key, orange is paper key and that gray is if you were booked.

What this shows here is that each link in this graph is showing either a proof from the thing above it. This pink box here is a device, it's signed in his Github proof and his Twitter proof and then it also provisioned the PGP key, which then provision another device which then provision another device below that. When the Keybase client is checking the validity of the PGP key of all the way at the bottom, it's actually running through a sigchain making sure every link that signs in all of these various things is legit, they exist on the social networks all the way down the bottom and then they can know that whatever message came from this PGP key definitely came from Jeff Maxim here.

Another feature of Keybase is the generation of something called a per user key. This is important for when we're dealing with chat in the file system and every user on Keybase gets a so called per user key that's a device independent. This is a NACL key, this is not a PGP key.

This is the more modern system. This thing is also written into your sigchain. It's useful if you want to be able to encrypt something for all of the user's devices.

If you want to send – if you want to send me a secret that I can read from any of my devices, you could use one of these per user keys that's generated on my device, so anytime I provision a new device it immediately gets this per user key, because the provisioner can basically give it to me. Per user keys are most useful for teams, which I'll get to in a second.

The last thing I mentioned about the sigchains is I keep talking about how the server for Keybase is an untrusted entity. The last piece of the puzzle was making all that work out is to take these sigchain data and publish it in a way that even users, not users of Keybase can go out and verify that things that the server has not done anything weird like try to roll back, or try to remove people's signatures, or add new signatures or anything like that without a user actually doing it.

The way we do that is you can summarize all of the user sigchains on Keybase by simply hashing every single possible link that you have. Problem what that is it takes forever. If you'd have to go through millions of links and hash them all together, so the data structure called a Merkel tree, which is also using Bitcoin if you're familiar with that, where you can summarize very quickly a hash, a single number of the entire system fairly quickly.

What we do is we take that number and we actually will do a special Bitcoin transaction with that, what we call the Merkel route and effectively publish it into the Bitcoin blockchain. Anybody that has a full node running the Bitcoin blockchain there'll be transactions in there that represent any change to all the signatures on Keybase, which is neat.

Another fun part of Keybase and this is one of the things that really separates it from a lot of apps like Signal and things like that is the ability to form teams. These teams are more than just an entry in a sequel database, or something like that says these are the people on my team. Teams also have sigchains that represent all the additions and subtractions of people into the team and they're all verifiable by the Keybase clients.

The purpose is for these teams to have either your company use it, or you can have your friends, if you have some group of friends you want to have talked on here you could do that as well, or your entire company could be on there. All of our applications like chat and the file system work with teams, so the picture on the right there is a picture of our chat, which runs on the mobile app, as well as a desktop.

Moves to be a little bit like Slack, so people with a combination of Slack and iMessage, I suppose, that's the team situation. The way the teams work is using these per user keys. It's basically there's one secret for the entire team, that secret is disseminated to people using all the per user keys from the people in the team. The Keybase server can't just add people to the team, a client has to do it and on so on and so forth.

You also have sub teams, something Keybase.board within a team, which can have members of the team, but cryptographically is separated, has different team secrets than the host team. This shared team key is automatically to set – roll basically whenever anybody revokes a device. If somebody revokes the device, you want to make sure that that person can no longer access any future communication within that team, and so everybody rolls up to the new version, it's sent out to everybody again using this per user key.

I started off this talk and talk a little bit about how there's useful apps that are built upon all this infrastructure. Now we have a system, does a  pretty good job of identifying people on the internet that you know, giving you access to a variety different public keys, either their PGP key or these device keys that we generate from the Keybase software.

What can we build on that? Well I touched a little bit on, we have some – the chat application and the file system are two of the main ones for us. Key requirement though is that these things are real apps. Part of the reason PGP fail is just – it didn't have a great app experience that people could really get involved with and use.

Our apps try to be just as good as their unencrypted counterparts, like chat should be just as good as Slack, or at least close to it, so that the people that are using it don't forget about the fact that they're actually using a secure app and only focus on the fact that they're just using the bad version Slack.

That's a big deal for us. Keybase chat is really one of our – probably one of our best deployments of an application built on top of the Keybase infrastructure, totally unencrypted, you can identify people simply by their social media and it has its own fully integrated with teams and runs on every single platform of note, all the desktop platforms; Linux, Mac and Windows, plus Android and iOS as well.

Unlike Signal or a lot of these other apps, there's no need to get in some out of band secret transfer. You have to take a picture of a QR code, or get together in exchange a number or something that pops up on your screen everything is handled through the Keybase ID systems like the identity system we just talked about.

Here's just another shot of the chat screen here. All these messages are signed and everything else. One cool feature of Keybase chat and teams in particular is you can have an open team, so these open teams anybody can join. If you're running an open source community or something like that and you want to run it on Keybase chat, you can do so. You can create a team and set it to be like anybody that wants to get in this team, gets in this team, and then you're in.

I mean, I ask what's the purpose of that with an encrypted app, but at least when you get people in there everything is signed so you can know who's talking. If you're running this thing and you're speaking in there, everyone could know it's truly you. Furthermore, once they're in there you can exchange DMS and things that with people and that's fully encrypted as well.

We have one team on there for a new cryptocurrency called Chia, which is like a environmentally friendly version of Bitcoin. They have a team on here of 1,800 people, so that's been our most successful team yet. You can also join the Keybase friends team, which has about 1,300 people and talk about Keybase up in there if you want to.

The other main application, I'm almost done here is the file system. This works a lot like Dropbox, the infinite mode of Dropbox. The traditional way Dropbox works is you just have a folder on your computer and the Dropbox thing will go and find what change and upload to the server, but all those files exist on your disk.

With KBFS, they don't. You're effectively just communicating with an API, and furthermore everything that you're doing is encrypted as well. Again, this works in pretty much the same way; everything in here is encrypted and signed and works on all the platforms, except for the phones, but the phones are coming soon, and you get plenty of free storage.

The way KBFS is laid out is you use your Keybase handles to get in there. My private folder here if I'm on a Mac or Linux, I can just go to /keybase/private/mikem and all my files will be there. Now they're only encrypted for me and all of my devices that are associated with my Keybase account. Not my PGP key, but if I have some set of device keys, all of them can get in there and see those files.

I can also share files. I can go into Keybase private and then somebody come on MikeM. If I want to share with Jeff Maxim again, I can stick a file in there and now everything in there is encrypted for all of my devices, as well as for all of his devices immediately. You can also do something cool with Keybase where you can share with a social media handle, so I can share with MikeM, some name at Twitter. What that'll do is something cool or it'll create a new folder, it'll have files in it and as soon as somebody proves that Twitter account, they get those files. We call this a sharing before signup thing.

If I know somebody on Twitter, I can go and put a file in this path right here, MikeM, Zem at Twitter and then contact them somehow and say, "I got some files waiting for you on Keybase. All you got to do is prove that you own this Twitter account," and then they get them. You could do the same thing with chat. It's an interesting feature. You can deal with all things. You could do that with a PGP fingerprint, you could do somebody at PGP fingerprint, you could do something like that with a DNS entry, it's very flexible.

The last thing I'll talk about here is there's the final application of Keybase, which is encrypted git. We provided this service on top of KBFS that allows you to push a git repo and set Keybase as a remote. You can set like, you use this fancy URL here at Keybase:// and then the location in –KBFS we're actually going to push this thing and it will go before the files land on Keybase servers all be encrypted with whatever device is doing the push to that repo and Keybase can't read them.

You could do that with teams and personal, so if you have a team you can create a team repo and if you're a dev ops team or something like that, you need to share secrets, you can push them in there and nobody can see them except for the devices that are signed into whatever team has access to that git repo. It's cool. It works pretty well in terms of with conflict resolution and all stuff integrates with popular git tools and things like that.

It's a neat feature. Like I said, it's very useful if you want to share secrets on your team, like I said like AWS access keys. I'm not going to recommend this right away for that, but you can think of secrets that you can put in there that would be nice, that's encrypted git. That's it. That's it for the talk.

**[0:32:37.9] JM:** Great, so now we got a little time for questions. Does anybody have any questions to start with? Question is does that Keybase git thing work on mobile?

**[0:32:46.5] MM:** You can view the repos on mobile, but you can't do anything, like you can't check them out, or you can't push to them, or anything like that, but you can you just see the list and you can see this list basically of which repos you have access to, that's it.

**[0:33:00.0] PARTICIPANT:** How do you make money?

**[0:33:02.3] MM:** The short answer is we don't currently have any revenue in the company right now. The goal of of Keybase has been to get as many people involved with it as we can. I think that the chat application in particular has a nice network effects going on with it. The more people you can get chatting, especially in a lot of these open teams, the more they can get their friends in there, as long as the app works well, we're pretty confident we can get more and more people in there and start using it, Keybase gets more and more popular. That's really what we're hoping will happen.

I think there's been a lot of discussion about how Keybase could make money. There's been talk about teams of a certain size, or corporate teams could have to pay similar how somebody Github Enterprise works or something like that, or Dropbox. That's one option. You could charge for the KBFS base, that sort of thing.

I mean, Keybase has been fortunate that we've had plenty of support from people looking to fund the company. I mean, Keybase had a pretty good round when it first got started, and then we recently entered into a nice agreement with the stellar development foundation. This was just made public a couple weeks ago, where they are now helping to support Keybase as well.

For the foreseeable future, Keybase will continue to exist, your files are safe in Keybase. They're not just going to disappear and that sort of thing, because Keybase goes out of business or anything like that. I think right, like I said to summarize, we're thinking more about growth, getting people using the platform and then we'll worry about – there'll be ways we can figure it out.

**[0:34:32.2] JM:** I'm going to ask you a question that's tangential to what Shawn just asked. With Bitcoin private key management today, you've got two polar options; one is you manage your private key yourself, and if you lose it then you lose all your Bitcoins associated with that. The other polar end is you have Coinbase manage your private key. Could Keybase be something that is in between those two options along that gradient?

**[0:35:01.9] MM:** That's great question. Yes. I think over the course of the coming months, we're going to start to see that. I mean, I think that's something that were going to be working on not with Bitcoin, but with likely with Stellar. Stellar for people who don't know is a payment network that has a cryptocurrency associated with it called Lumens. It's a lot like Ripple if people have heard of that. That's what we're hoping for.

I think one of the nice things that Keybase can bring, one of nice advantages that Keybase can bring there is that if we do it right, we'll be able to – your Stellar wallet will follow you around from all your devices, so you'll be able to for instance be able to send payments from your phone, or from your laptop, and the key, the dissemination of that of whatever your Stellar private key is will happen with – Keybase will do it, so you don't have to worry about transferring it yourself, similar to just how you don't have to worry about transferring any of the other keys associated with your with your Keybase account.

Yes we're hoping that – we're hoping that Keybase will really make it possible that you can have convenient cryptocurrency payments without having to have your private key on a third party like Coinbase and without having to carry a hardware wallet around wherever you go.

**[0:36:10.0] PARTICIPANT:** Testing. My question is what does the playing field for this type of product look like as far as other competitors that are doing a similar thing, or could in the future try to attempt to do a similar thing and what other types of things could they do to compete and vice versa?

**[0:36:28.5] MM:** Yeah, I mean, I don't know if there's any real direct competitor to this particular idea. Certainly there's competitors in the various apps that we're actually working with here, particularly chat. I mean, there's a lot of encrypted chat applications, many of which are more unpopular than Keybase, like signal for instance.

I think a lot of what we're competing against is the existing applications in this space as well, things like Dropbox and Slack and those sorts of things. We want to be able to convince people that using our application has real value, that these crypto tools, I mean, this is becoming more easier by the day as more and more privacy related problems come out in the world, people are more and more looking for this solution.

I think in terms of other ideas to the identity problem, there's been a variety of ideas tossed around about blockchain solutions to the problem. Can you build something on top of Ethereum for instance that some smart contract that somehow encodes people's identities in a similar way the Keybase does? I think a lot of those ideas run into to usability problems, like any decentralized application always comes up against performance problems, comes up against just bugs that are very difficult to find, very difficult to fix, because you have to disseminate your software to everybody that's running this thing.

It takes Bitcoin sometimes gears to get changes to their core software out to the wild fully deployed, just because they need everybody on it. I think for us, a lot of it is just carving out what we think is a new space here, where we have these crypto applications that are good, that that are easy to use, that are hopefully somewhat easy to understand and that people want to use and go from there.

Then hopefully that they provide a compelling value at over things like Signal and more specialized apps, just because of all the things they provide and this identity model, which we think is pretty novel. Does that answer your question?

**[0:38:23.8] PARTICIPANT:** Yeah, it was awesome, Thank you.

**[0:38:25.9] PARTICIPANT:** How does account recovery work?

**[0:38:27.7] JM:** How does account recovery work?

**[0:38:29.6] MM:** That's a great question. It's actually we spend probably about 40% of our time working on making that easy. Basically, the way it works is there's an option on the site where you can so call reset your account. You're only supposed to do that if you lose access to all the devices that are associated with your account. For some reason you lost your phone, your computer and whatever else and you're out, then then you need to reset. Now big bummer of having to reset is that you lose all the information that's associated with your account.

If you had data and KBFS, or if you had chats, they're gone as soon as you reset. You're effectively like a new person. Typically way we do the reset is you can if you still control the e-mail account associated, we will send you a new password and you can reset start over again. Or if you've completely lost it then you come to us so we go through some customer support thing to make sure you are who you say you are deal.

The stakes are definitely higher for losing these devices as opposed to something on Dropbox, but they can probably get you your data back, or Slack where you can probably get back in to your team. With Keybase, it's very important for us that you not get back into your team, because if you've lost all your devices, you're really no longer you as far as we're concerned.

**[0:39:40.7] PARTICIPANT:** Do you have any thoughts on tattooed-based paper key recovery?

**[0:39:46.2] MM:** Yeah, well not any specific thoughts. I mean, that's a good point. I didn't really mention paper keys too much, but there is like, you can create a paper key, which is effectively just a giant pass, giant path like a bit 20 pass for your site thing that you can used to recover your account. If you lose that too though, you're again out of luck.

**[0:40:04.8] PARTICIPANT:** That's why I figured like a QR code in a not visible place.

**[0:40:07.7] MM:** Yeah, if you show up to the office, maybe that would work.

**[0:40:13.2] PARTICIPANT:** I guess, this is related to some of the wallet stuff you guys were talking –

**[0:40:17.1] JM:** Speak a little closer.

**[0:40:18.1] PARTICIPANT:** Sorry the wallet stuff you were talking about before. This is crazy, but I'm wondering you were talking about sharing maybe git or sharing – having KBFS to share with specific people, or as part of a team. I'm wondering if there are any plans to have specifically sharing of secrets. You mentioned AWS access keys in the git context and how that violates your impulse, like you wouldn't want to put that stuff in git.

**[0:40:45.0] MM:** If you want to make any specific security, right?

**[0:40:46.6] PARTICIPANT:** Right, right. Yeah, we don't want to touch that, but I'm wondering if I have an API token that I needed to share with my team or some personal account information that I was sharing with family, are there any plans to have that – I mean, you could do it with git or KBFS, but it seems something – I guess, a Keybase version of Vault or LastPass or One Password or something, I guess is what I'm asking is anything like that plant or just use those products?

**[0:41:13.7] MM:** My best answer to that, it would be you should just use KBFS, I guess, or chat. I think one thing you can do is you can have – if you feel comfortable, you could put this in chat, or depending on how long you want to get at, if you're talking about something that should last for a long time, I would say KBFS is probably going to be your best option. The good thing about KBFS is like I said, it's available on all your devices. Currently it doesn't really work on mobile right now. In fact, doesn't work at all. You can view what folders you're in, but very shortly I mean, we have this thing – I mean, our prototypes we're running it right now where you can get all your KBFS files on your phone as well.

All this stuff should follow you around. I think more specific like UI is things like LastPass, or – people always ask us, it's like, "Why doesn't Keybase have a password manager?" For instance, it seems like a natural fit. One of the reasons is because a lot of the challenge with those kinds of apps are actually UI challenge. Why is One Password so popular compared to the others, is because it's got a great UI. It works really well if people know how to use it.

For us, if we wanted to do something like that, we'd have to become experts on how to really make a good UI like that. Keybase it's built with third-party integrations in mind. We're hoping that if Keybase gets to the point where it gets very popular, that people will start building tools maybe like that on top of it. If you had a particular use case that just wasn't really fully covered by KBFS, or git, or wasn't covered in a way that you really liked a password manager or a note system, it sounds maybe that might be what you want, that somebody could maybe build one.

I think that that would be the dream of Keybase. That's how I think we would know that we really caught on if somebody actually did do something like that and whatever they built became popular.

**[0:42:55.1] JM:** All right. Michael Maxim, thank you.

**[0:42:57.9] MM:** Thank you.

**[0:42:59.2] JM:** Okay, so we're going to take a break until 8 p.m. and you can get some more pizza, get drink, go to the bathroom, all those requisite things and then we'll be back at 8 p.m.

**[0:43:11.6] IR:** Scratchers we put out on your seats, they do have those free passes to Dash.

**[0:43:16.5] JM:** All right, so this is Habib Talavatifard from Clarifai. I have had the CEO of Clarifai on the show for and it was an incredible show. Clarifai is basically a machine learning, computer vision API, I think if that's a good explanation for it. One of the things that we discussed on this episode, the episode we did was an overview of Clarifai, so it was an overview of this machine learning computer vision company.

One of the things that he discussed was the process of training and deploying machine learning models. The deployment cycle and the retraining of machine learning models is pretty underdeveloped area of software engineering, I think. I was really happy to have Habib be willing to come to the meetup and I'm pretty sure that's what he's going to be discussing.

**[0:44:09.8] HT:** Hello. Thank you very much for have me here for this meetup meeting and thanks Datadog for hosting the event. I'm Habib Talavatifard. I'm a technical lead at Clarifai. I am in charge of a product called visual search and also another product called custom training, but today I'm going to try to actually talk a little bit about everything, including infrastructure so let's see how well that goes.

I guess, that's the agenda for today. I'm going to talk a little bit about what our company does and then I'm going to talk about the stack for the duration. About Clarifai, our history is that we started at 2013 after our CEO, actually I mean, she was a PhD student at the time, one image that you know contest, which is actually a pretty important contest in image classification.

Then he founded the company at that time based on the research that he was doing at the time. Since then, we actually have grown a lot. We have build a lot of things. For now, we have a variety of products that we have. We have machine learning as a service, as an API, we also have machine learning solutions and we do a variety of services that I'm going to probably touch on two of them in the next slides.

I guess, that's the list of our investors. I guess one disclaimer here. I'm going to talk about GPUs a little bit and GPUs are actually machining learning are tied to Nvidia. Nvidia happen to be one of our investors, but that's all coincidental, I'm not speaking for Nvidia or anything. What we do; part A, models.

We actually build these previewed models based on convolutional neural networks that you could actually use from our API to actually classify within different domains. We have a general one, we have code, no moderation and SFW all kinds of stuff. Most of them are based on this idea of no convolution, or of this course, or deep learning. I guess you probably have heard that there is a lot of recent advancement in that area, and we're trying to actually bank on that.

If you go to our website, you might actually find something called modern gallery, which is a bunch of models that are available. We have all kinds of stuff there. For example, one is celebrity, the other one is demographic, all kinds of stuff. General model, getting embeddings and all that.

I'm going to talk about one of them just to give you an idea how well these things actually work in certain cases. We have an NSFW model that actually is pretty good. Why I claim that it's pretty good, because that's I guess the performance curve of this model. If you guys are familiar with this graphs, that's exactly how you want your model to be. Not having much true negatives, well you have very good true positive, which means that you have extremely good precision and recall.

This is an example of the models that we actually provide. A lot of customers actually use this for moderation purposes. For example they want to detect that if they have some user posted NSFW content on their website, they can actually detect using this mode. I want to show you some examples of how well this thing actually works.

Look at this one, I have lags. This thing is predicted as if forward, the probability is actually pretty convincing. Also you can see the other one, the general model that we have and it's actually predicting a bunch of tags, and you can see is actually pretty good. It’s bathroom, bath tub, bath shower, wash, all kinds of stuff.

This variety of tags, or [inaudible 0:48:03.2] that actually produces is very useful. You can actually use it for other applications, including search. Here is a bunch of our examples. This one also was predicted as 0.97 safer work, which is what you want. Another one, 0.99. 0.99 safer work, 0.99 safer work. Yeah.

Yeah, so this is an example of the model that that we build and be making available to our customers, or actually two examples, because you see the general model as well. Let's actually move on to the second thing. Actually one thing that I personally am pretty much involved, I'm actually in charge of this particular product, which is search.

Store runs your data. A lot of people actually, what we realize is after they use our models to actually tag their data, they also almost all the time do some search on top of it. We figured that maybe we should do the search to help these guys. We then tell them and do it, do it. We can do all kinds of searches, search by tag. If it look something like this, basically you can search by okay, I added all my data, I want all the dogs, there you go.

You can't do something like a reverse image search. Basically you search by an image and we find the images that are similar to it, and we even can combine different searches, so I can say dog and that image and then it finds these things for you. The dogs that are wearing red shoes, so words descending okay.

This is the second product that I wanted to give you an example of this stuff that we do, but let's move on to the second part of the talk, which is basically our stack. Although I have to mention that actually we do a lot more than this. We do all kinds of models, all kinds of stuff, just wanted to give you a taste. Our stack, I guess I'm going to dive into it. I'm going to show you a diagram.

That's what we do. We have an API and this is actually the API that we have on our production setup. Most machine learning companies actually have usually have a production setup and a experimentation setup, which I'm going to talk about more. This is basically our production setup for the API. I'm actually simplified this to the almost non-existent, so I'm just keeping the most bigger pieces. Yeah, what's up?

**[0:50:45.0] PARTICIPANT:** The start right, so you would be searching within your database. If I were to use your API, can I upload my data and serve –

**[0:50:57.7] HT:** Yeah, Yeah, so the API supports indexing. Basically that is your data. Actually what I showed you is you actually –

**[0:51:05.7] PARTICIPANT:** I include them in –

**[0:51:07.0] HT:** Yeah, it goes to our servers, it gets indexed, it becomes available for search, like any other search services. That's something that I have to – so basically, you can actually get that on our servers, which are located on AWS right now, but one thing I want to mention is actually one of the challenges for us right now is trying to actually decouple our self from the cloud platform.

We want to actually not be tied to any a specific cloud platform, because a lot of customers, one specifically for those platform, or don't want any specific platform.

**[0:51:43.1] PARTICIPANT:** On the degree, like these are special code to list the data.

**[0:51:46.9] HT:** Yeah, and then you – it enables you to actually support on prem solutions as well, which is another set of solutions, which is also pretty important. Basically, I want to talk a little bit about this guy. In the production setup, you have API layer that basically is behind a load balancer, all the usual stuff, encryption all that, you can index your assets and they are going to be actually want to a bunch of machine learning models.

Once they are run through those machine learning models, the result is going to be written to the database. Machine learning models can produce tags similar to the stuff that I just showed you, but they also can produce something called embedding, which is a representation of whatever concept that you are trying to do according to that model. Those things, you can actually store and search against. That's what we are doing.

Also, there is a read pass, which is doing the search query and basically that already consults the database, but it actually is not the simple lookup. What you actually you do usually in these cases is you actually try to do the search as mark me, by clustering your data and doing a two-stage process. Basically when you do the search, you first actually match and find the relevant cluster of the data, you fetch candidates from there and then you do a second search on the candidates.

It's a multi-research process and you probably don't want to just brute force it, because it doesn't work at all if you do that. It won't scale. That's the gist of it. Now for the database part, since this is an infrastructure talk, we actually use Citus MX, which has historical reasons, plus also is something technology that's useful in our case.

The original version of this system was written using RDS, basically PostgreS, single node, whatever. At some point, we wanted to a scale it and I personally didn't want to write everything on a different technology. I decided just scale it using something that looks like PostgreS. Citus is actually a shard of PostgreS essentially. It has a lot of nice properties, for example it has a ring architecture, which means that everyone can be master, which means that when you do a query, you're going to actually hit any of the nodes as the coordinator, or master of your query.

That means that your read and write throughput is actually multiplied by the number of nodes that you have in your machine, compared to RDS, which is extremely useful if you are trying to build a high throughput system. Also, it actually helps us to serve the search queries pretty fast, because actually paralyzes the result like any sharded system, so we can actually search multiple shards in parallel. That's why we use it.

It also has nice transactional properties, which in this system is actually useful, because this also, in addition to search, those other stuff and we want to have certain transactional properties. For example, Citus MX recently implemented distributed transactions, which I guess if you guys have heard is pretty difficult to implement. Maybe only a span area and cockroachDB who actually successfully implemented that and Citus MX has, which is very good for us.

Yeah, I guess the noteworthy stuff, I mentioned a bunch of them. I guess we use Go and most of the non-scientific computing part, for the scientific reading part you just use Python like any other ML shop. We use GPUs at inference time in this stack, we also use GPUs at the training time, which is not shown here, but that's a different story. Basically Citus helped us actually to scale our rights, so that's another thing.

The most important thing, which is the last part of the talk is actually you are using Kubernetes to actually manage all the containerized application services in the previous slide. Everything, you can see that there is a section application services and there is a section data services.

Everything on the application services is authorized and manage microwaves, API layer, another stack indexing.

Data services are outside of the Kubernetes and they are either on AWS, can be a cache, can be Citus, which we use — manage Citus. Basically being a startup, we don't want to actually manage our own databases. They are outside Kubernetes. Inside Kubernetes, the MLS stack uses GPUs and we want to actually talk about that and it's a good one.

Oh, before that, actually one of the things that this stack has is actually monitoring. We extensively use Datadog and I think I've heard that, that actually we were one of the first users of Datadog for Kubernetes, so apparently we started at a time that they were all running bugs and it was before my time, but apparently those days were very interesting. You're using it and we're pretty happy with it. This is an example of a dashboard that we have.

Okay, now the last part of the talk, which is the GPUs, which is most more infrastructure. Show of hands, how many of you guys actually have used GPUs in production? Okay, so good. I guess, I can talk a little bit more about the details. Basically when you are using GPUs for machine learning and production, usually you have two kinds of use cases. One is actually some batch training, or some batch processing, which is usually for training. Then there is a real-time online use cases that usually happens at inference time, basically when you are trying to use the models that you just trained.

We actually use GPUs for both use cases, but I'm going to focus on the second one. Very recently, we actually managed to use Kubernetes for handling the first use case, which means that we actually – our experimental set up also runs on Kubernetes if we want to. We can actually scale our infrastructure for training to hundreds of nodes and then scale it down on notice if you want to actually do a quick experiment very extensively.

That's all I'm trying to talk. I'm just talking about using the mesh GPUs, add inference time in that stack that I just showed you. Which is basically what I said, then Clarifai workloads are either for training, or on that force, or prediction and inference. Basically, we run Kubernetes on AWS, which is what we do.

Now everything is nice, but I want to actually talk a little bit about how we actually end up using the Kubernetes. We are going back maybe to 2015 November something. At that time, our company we're running on virtual machines. Deployment meant that actually you have to build these virtual machines, build the new set, move the traffic to this new set of virtual machines and destroy the old one.

If you did a small code change, it would mean that you need to spend several hours just actually just working on this thing and banging your head on the keyboard. That was the old system. It was very difficult to work with. Then our infrastructure team actually came in and they decided to use Kubernetes. By the way, disclaimer so I'm – like I said, I'm not a member of the infrastructure team, so I'm actually basically presenting their work. This is not Microsoft.

They said we want Kubernetes, we want containers, we want discovery, we don't want BMs. I remember, this is November 2015. I guess in theory, theory and practice are the same and practice are not. In practice, they run into problems, Kubernetes at the time had a lot of useful features, but it actually lacked – It's not.  No GPU support, no SS support, no SSL support the ELB, no ECR support.

Easy to get your whole AWS account rattled. That one is actually interesting, so basically the way Kuberntes works, they actually try to – ask AWS or what nodes are open, what's going on and what your metadata and all that. At the time, you just could easily get yourself in all trouble because of series of bugs on your entire account, because you made too many of these requests.

Then, I don't know, you were developing something on your test cluster, suddenly your entire AWS becomes struggle. It was actually extremely painful also to debug as well, because what's going on. Our company at the time, we ended up actually submitting changes to open source and Kubernetes at the time to address all of these issues.

We added the support for GPUs some form of experimental version of it. We added as a support for ELB, ECR support and all support with AWS to make sure that that thing doesn't happen totally, and make sure that that metadata is actually cached on the Kubernetes side, so

basically nothing bad happens. I'm going to actually more go – I'm going to skip the last three and I'm going to talk about the debate about the GPU part of it.
The GPU part of it, November 2015 how do you get GPU support? You find expert engineers. I guess, you guys probably know board, which is a product that equivalent to Kubernetes that actually Google uses internally. It has support for GPUs for years. You find these guys as GPUs over in Kubernetes and then you profit.

There were issues, realities, stuff like that. We had long discussions about how to do it essentially at some point, but the problem is in these projects if you actually add something willy-nilly, you might actually bridge future releases and something bad happens and it's hard to predict that how things evolve. We really want to actually get it right and just not randomly add the stuff.

Essentially at some point, they come up with a very hair down basic version. It will become available at July 2016. It was nice, but we have limitations, actually lots of them. You could have any parts that you want, as long as it’s by Nvidia. Actually, the way you actually make this thing work, this flag that they just put here experimental, Nvidia GPU, you have to actually add this to your kubelet run command. Basically that's how you get it.

It only supported one device per node. You couldn't tell that what GPU you are using. Is it an ancient one, or is it something new, or how much off the RAM of this thing it was using? No resource usage reporting, nothing. No data log for that matter. We still did it. Also even more revision. This this one is actually pretty awful.

This thing actually for it to work, it needs actually certain kernel driver libraries on the node itself. You cannot package your libraries into your container and ship it. It won't work. You actually have to make sure the correct libraries are actually put in the node, which is really painful. If we don't, it will crash with the most awful error or like unhelpful error that you have seen and you cannot figure out what what's going.

You have to deal with that, so it means that you have to – when you were actually managing your nodes, you have to make sure that you have the correct libraries, which usually people don't do in Kubernetes work, but yeah, we were doing it. Yeah, they did it and it was useful for us. Why? Because that three hours that I talked about, that would take us to actually deploy a code, after this was paired down to five minutes.

We could actually scale up and down by just clicking the dashboard, which is was great for us. In the other stack, those ML machines, ML models that I showed you, you could just scale up and down those things on-demand at moment notice. Recently we actually added auto-scaling, so you can actually auto-scale automatically based on traffic and stuff like that, which is pretty nice.

Now we added all these things around 2016, but these days things are much easier. You don't have to deal with a lot of these issues. For example, recent developments, there is this concept of device or hardware plugins that Kubernetes introduced, which essentially abstract away the problem of actually where the system libraries are, or any other problem related to the device for you.

For example, what kind of Nvidia card you have, how many of those things have, the stuff like that. Instead of actually somehow hard coding this somewhere, you actually just ask this device plug-in that how many of those guys, how many GPU you have, how many of whatever you have.

You get to know the libraries passes and everything, which is nice, because it means that whatever system that you design will be portable across different clusters. Multi-GPU support added, which means that you can actually use more than one GPU in each part, which is great and actually be needed, because a lot of our models don't fit in one GPU. Also there is this demon set that actually helps install – there'll be the installation of these drivers, so we don't have to do all those manual awful things that we used to do.

Actually, we use these all these things in our experimental setup that I mentioned, that we can scale to hundreds of machine nodes if you wanted to, and we use these guys. Still, I just mentioned was works for us, why do it.

Yeah, and I guess there's also, the battle is not won. There's a lot more to be done. For example, we need better scheduling, some advanced scheduling that basically knows an average of what car do you have, what's the limitation, stuff like that and schedule based on that. For example, something like affinity is very important in this work. If you have a node before GPU cards talking for example from cart 0 – the communication between cart 0 and 1 might be actually faster because of affinity issues, versus 0 and 3. You really want to actually get 0 and 1 when you ask for 2. You don't want to get 0 and 3. That's an issue that has been around in Kubernetes.

I think it's not limited to GPUs, CPUs also, but we want this smart scheduling, which is something that useful for us. Another thing that we really want to see in Kubernetes is actually device sharing, or overcommitting, which by the way we actually do right now. There is a hack, or trick that you can do that you can actually get this, but I think with the newer version of Kubernetes, our hike will break and we want actual official sharing and stuff like that.

The hack is actually quite interesting. Remember that I showed you that flag that you can say one Nvidia GPU is equal one, actually nothing stops you from saying I want a 100 GPUs on a machine that has only one GPU. What it does is basically creates a hundred objects and it maps it to one GPU, so now if one application comes in and says I want 20, it's going to give you 20% of that 100.

It's basically some form of sharing, which is pretty useful, because a lot of models actually don't use the entire GPU. You can actually run a bunch of the smaller things next to them, so you end up actually using a lot less GPU in your in your machine, in your cluster with this thing. The problem is there is no enforcement. If some process that you're running decides to suddenly do something nasty, nothing stops them from actually hogging all the memory on the GPU and causing huge problems for everything else that is actually being run there.

You can only really run, do these things in an environment that you actually trust other processes, which is actually the case for us, because we can actually predict how much memory each one of our models actually need, because they usually allocate at the very beginning and that's it. Then they're just flat still. Yeah.

That's a story of the adding GPUs. These days, people actually, I guess very recently Kube flow is added. It's a lot easier, but I guess we were the very first company that actually started using this and we look forward to actually even, use it even more, and it's an internal part of our business.

**[1:09:08.2] JM:** All right. Habib Talavatifard. Questions?

**[1:09:13.9] PARTICIPANT:** How do you manage Kubernetes in AWS? Do you do it ourselves or do you use –

**[1:09:18.5] HT:** We use ECR. That's why we added the support for this ECR basically.

**[1:09:23.8] PARTICIPANT:** What did you say? E?

**[1:09:25.1] HT:** Yeah, that's the container or something. That we go. Registry. Yeah.

**[1:09:30.3] PARTICIPANT:** No, I mean, Kubernetes itself. For example the Kubernetes worker, do you guys install and provision the Kubernetes workers yourselves?

**[1:09:39.2] HT:** Yeah, yeah, yeah. We actually provision the machines ourselves manually.

**[1:09:44.6] PARTICIPANT:** Cool.

**[1:09:45.4] PARTICIPANT:** Do you provision bare metal servers for the GPUs to run efficiently, or how does – do you provision VMs?

**[1:09:53.1] HT:** You can do both actually. There are cases that you want to do VMs. For example if you have an experimental setup. You can actually give the entire experimental setup to the Kubernetes and then no one can touch anything, but some researchers or AML machine-learning engineers will be not very happy that they don't have direct access to the GPUs. One thing you could do is you could actually run two VMs and give half of the GPUs and resources and everything to one of the VMs, and give another half to everyone and then run Kubernetes on the second VM, while if the first VM is actually kind of while this, everyone can go for it, stuff like that.

**[1:10:35.8] PARTICIPANT:** One more unrelated question. How do you do the feedback loop of once you classify image, how do whether it's right or wrong? How do you get that feedback loop and sort of –

**[1:10:46.0] HT:** Yeah. Our API actually has a full evaluation suit, which you can actually run on the models that you customize yourself. One thing that is supports that I didn't mention or plus, or you could actually be able to cost your own custom mods and you can evaluate those. That's one part of this evaluating those.

The models that are pre-built and you cannot customize, like the general model NSFW, we actually pretty much do a lot of experimentations and evaluations before we shoot them. All the normal stuff that everyone does, having test sets, looking at the different metrics that you want to actually have in a multi-label classification.

Actually I showed one of you the chart of true positive, versus true negatives. You look at all those metrics and only then those things are actually good enough across the board, we actually ship those previous models. Interestingly actually the code that we use to evaluate those previewed models is the same as the code that is actually in the API who's exposed to the users. They get work on the same principles on everything.

**[1:11:58.8] PARTICIPANT:** What's your container format?

**[1:12:00.4] HT:** We use Docker basically. It's probably amazing.

**[1:12:03.1] JM:** Any other questions? All right. I think we're good. Thanks Habib.

**[1:12:06.9] HT:** Thank you.

**[1:12:10.5] JM:** Now we'll hang out for another 15, 20 minutes and chat or take off whatever you like. Yeah, I want to thank the speakers Mike and Habib and thank Datadog of course for hosting this awesome meetup, and all of you for showing up, because this was an awesome community, really great conversations. Thank you all.

[END]