EPISODE 1111

[INTRODUCTION]

[00:00:00] JM: Across a company, there is a wide range of resources that employees need access to; documents, S3 buckets, git repositories, many other things. As access to these resources changes across the organization, a history of the changes to permissions can be useful for compliance and monitoring.
Indent is a system for simplifying access management across infrastructure and it allows users within an organization to request access to resources. It keeps logs of the changes of who can access those resources, and this is useful.

Fouad Matin and Dan Gillespie are the founders of Indent and they join the show to talk through the application of access control management as well as the architecture of Indent itself, which has numerous interesting engineering decisions within the architecture.

Before I start the show, I want to mention, I am investing in software companies. If you are building a software company and it's targeted at developers or it's an infrastructure tool, I'd love to hear from you. Send me an email, jeff@softwareengineeringdaily.com. Also, we are looking for writers. If you are looking to write for softwareengineeringdaily.com, send me an email, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[00:01:17] JM: GitLab Commit is coming to London. GitLab commit is GitLab's community event. GitLab is changing how people think about tools and engineering best practices, and GitLab Commit is a place for people to learn about the newest practices in DevOps and how tools and processes come together to improve the software development lifecycle.

GetLab Commit is the official conference for GitLab, and it's coming to London October 9th at The Brewery. If you can make it to London on October 9th, mark your calendar for GitLab Commit. Go to softwareengineeringdaily.com/commit and sign up with code COMMITSED to

save 30% on conference passes. If you're working in DevOps and you can make it to London, it's a great opportunity to take a day away from the office. Your company will probably pay for it.

I am going to go to a GitLab Commit at some point. It won't be this one in London, unfortunately, but I will definitely go because I am excited about the GitLab ecosystem. It's quite an interesting ecosystem, and seeing it develop has been cool over the course of Software Engineering Daily's lifetime. A GitLab Commit, there are speakers from VMware, Porsche and GitLab itself. So if you're in London on October 9th, you can check it out. I hope you do check it out. And thanks to GitLab for being a sponsor.

[INTERVIEW CONTINUED]

[00:02:53] JM: Guys, welcome to the show.

[00:02:54] FM: Thanks. Thanks for having us.

[00:02:55] DG: Thanks for having.

[00:02:56] JM: You both work on Indent, which is a system for access control. Explain what access control is.

[00:03:03] DG: So, the way we look at access control is where within an organization, you have someone who needs to have access to something. And in order to ensure that the organization is treating their user's information safely, that needs to be closely guarded.

[00:03:18] JM: Okay. And so access control can be for apps. It can be for files. Explain in more detail what that means.

[00:03:27] FM: Yeah. So, there's a broad range of things that people need access to on a daily basis. If you're, let's say, on the sales team, you might need access a purchase order in box. If you're an engineer, you might need access to a cloud storage bucket. But generally, you only need that access for a few minutes at a time. Maybe just 15 minutes. And then you don't need it anymore. But the process of actually getting that access can take a couple hours, days, even

weeks. That takes you out of the work that you're doing and puts you in a waiting position while you wait for the approvals or any other compliance objectives to be met.

[00:03:58] JM: And isn't this handled by RBAC controls or Okta?

[00:04:03] FM: Definitely. That, you can kind of think of as the backend. That's the database, if you will, of who currently has access. The next piece is the change in management around it. Kind of like how you have a pull request, where someone needs to make a change and you might need some discussion around it to review to make sure this is an appropriate change and what the consequences of this change are. And that's the real piece that Indent solves.

[00:04:26] JM: So, what exactly does Indent do? What is the problem and the workflow that it's solving?

[00:04:32] FM: Yeah. In terms of how Indent works, people will submit access requests through a web portal, Slack, a custom internal tool that uses the indent API. Administrators can define policies, like who can approve access to an Okta group? Or who can approve access to that cloud storage bucket? And send the requests into, let's say, sack for faster responses. Then once we've received the required approvals, we'll send an audit event to a customer-defined web hook, HTTP endpoint, Lambda. And after the access duration is expired, we'll ping the web hook again to revoke the permission. And then finally, compliance teams can go into Indent to actually collect the audit evidence when they need to prepare data for their auditors.

[00:05:12] JM: Got it. Essentially, you have middleware set up to interface with all of the different things that access control could be required for.

[00:05:23] FM: Exactly. And our key focus is around the user experience. It's the end-user who is just trying to do their job or the approver who keeps getting an onslaught of access requests to make that process as painless as possible.

[00:05:35] JM: Remind me again, why doesn't Okta solve this problem?

[00:05:39] FM: Yeah. A lot of tools have some version of this. But, ultimately, it doesn't take in the complexity that companies generally have where you'll have an approval required from a hiring manager. You'll have an approval required from a system or a data owner. And then ultimately, you'll need someone in IT or security to actually scrutinize it. And that process takes a lot of cat herding. So it kind of is fragmented today where indent automates the process and help systematize all of those one-off interactions.

[00:06:07] JM: What's a common example of a piece of access control that I might want to interface with? Let's say I'm a developer. I need access to something. How does that work with Indent?

[00:06:23] FM: Yeah. Let's say I work on engineering team where we store data, customer data, in cloud storage buckets, and I need to help troubleshoot a problem for a customer. The usual process is all message the helpdesk, or I'll message that head of DevOps. Ask them to grant me access. Maybe they'll ask a couple questions and then they'll just grant me indefinite access. But after maybe 15 minutes or 30 minutes of work, I'm done and I don't need that access anymore.

With Indent, the developer would request the access for the 30 minutes they need it for. It would go to the right persons or manager, maybe the data owner. They would approve hopefully within a few minutes, and then that person could get that access just for the time that they need it for, and then we would revoke it. Really, focusing on that human-centric time bound access

[00:07:05] JM: This is an improvement in the workflow, because the manual workflow where I'm just messaging somebody, that's prone to all kinds of human error, right?

[00:07:17] FM: Exactly.

[00:07:18] JM: What kinds of human error could occur if I don't have a workflow management tool like this?

[00:07:24] FM: The most common is that people don't actually know what they need access to. They think they might need access as another example to all of compensation data in workday.

But in reality, they just are a recruiter who needs access to compensation bands in workday. There is some amount of discovery around what they actually need to access to. And then there's the time boundedness, where you're supposed to revoke the access after some period of time, but actually performing time bound access and approvals and revocations is very complicated and, generally, very frustrating if you're in the middle of a task and you're granted access for 15 minutes and it just automatically revokes. Indent really tries to focus on those end-user experience problems.

[00:08:02] JM: So, do you have any idea how access request management has changed in the past year? In the past several years? Because I imagine, there are some other middleware tools that have helped with this kind of workflow. Why wasn't there something like Indent before you guys started it?

[00:08:22] DG: I think that there's been a pretty large shift from having sort of enterprise-centric tooling where everything is built around like active directory and you've got – You're managing it within this Windows ecosystem. And I think the biggest shift over the last five years I would say a similar to cloud-based tooling, where you want to have SSO. Because the way that the previously – The legacies, almost legacy systems work, it makes it hard to work with the new class, which really means that like the surface area of access control has increased. I think that the problem area has increased to the point where organizations need something that can help facilitate that process to make sure that people are getting exactly the right amount of permissions within a timely fashion.

[00:09:07] FM: And when you look at large engineering teams, so any FANG, or really large funded startups with engineering resources to spare, they'll generally build an internal tool to solve this problem. That naturally just like within the internal tool, it can only take on so much scope. So after 18 to 24 months, companies generally have to re-implement that system.

[00:09:27] JM: Yes. This is the classic example of something that has been implemented internally a bunch of times and finally gets built as a cloud service. So there's also some advantages to having a dedicated system for this, it's because you can get an audit trail out of it. Explain how an audit trail works and why that's important.

[00:09:49] DG: Yeah. The way that we look at an audit trail is that we're taking a snapshot of every single play in the process of getting access, and that's important because it really does paint the picture and story of why and when people had access. And it allows you to go back and start reassessing whether they should have that access, or in the worst case scenario, being able to hold people accountable for what they did with that access. I think it's an essential part of any modern security footing, because you need to know what's going on and be able to at least have a sense that people can be held accountable.

[00:10:23] JM: Is this relevant for GDPR or any other common compliance situations?

[00:10:30] FM: Yeah. It fits into a lot of different compliance objectives. The privacy ones being very top mind and prescient, but there's also the long-standing ones like SOC 2 or SOCs compliance for public companies, HIPAA for healthcare, PCI FINRA for finance. Each of these compliance requirements have some sort of peace around either time bound access, reviewing the access on an ongoing basis, and ultimately moving towards this least privileged model. But actually implementing that requires a lot of manual effort. And generally, those teams have to throw headcount at the problem to try to fix it.

[00:11:04] JM: Is the onboarding cumbersome? Because I can imagine, if I'm onboarding with this kind of heavy middleware system, I've got to integrate with Okta. I've got to integrate with all the different things that need access control, like my AWS bucket, S3 access, my – Whatever. Kinesis stream privileges. I can imagine just an infinite amount of integrations that would need to take place almost to a point where I wouldn't even want this thing.

[00:11:31] DG: Yeah. Our goal is to try to make that as easy as possible for our end-users. So what we are doing is we're packaging up different solutions such that we can take on the large portion of the development that is required and make those integrations. And we're trying to do it in a way that is most secure for the user. Ideally, they are running weird. Just calling out to another piece of code we've written that's running on their infrastructure. So that way we never actually hold the keys that are used to make major changes.

[00:12:01] FM: And then in terms of the actual implementation, we'll provide these packed-up solutions like you need to send an email to a Zendesk, or to a helpdesk, or you need to control

access to an Okta group or control access to an IM group. Both those default out-of-the-box integrations that companies can even customize to work the way they want them to.

[00:12:20] JM: During a piece of access management, during a workflow of access management, what's going on under the hood? Is it just – You talked about in some detail that you have like the user makes a request. That request kicks off AWS Lambda. And then the Lambda function hits, I guess, whatever piece of access management software you're trying to automat. Actually, could you just refresh network flow, because I'd like to dig a little bit deeper into the engineering.

[00:12:50] DG: Yeah, totally. So we offer a set of APIs that either you call directly or you call through one of our existing integrations, like through Slack. Say, if you request an access something through Slack. That goes to our API servers, which then we're using a tool that was created by Uber called Cadence, which allows you to create workflows that sort of run like a Lambda, except for you're able to express relationships between them. And using this, we are able to have a high-level of fault tolerance while addressing these requests.

In Cadence, we're keeping track of the whole approval process and all the way from someone requesting to having the web book actually go out to grant access. In that, we are able to ensure that when someone is approved or when their access is revoked, that the request is actually sent out and that the receiving system actually acknowledges that.

[00:13:37] JM: That's amazing. We did a show on cadence not too long ago. And this is very informative, because it is useful for these long-lived workflows. If I recall from that show, basically, the idea is like this concept of a workflow where you have stateful, long-lived, multi-transaction kind of user sessions. The example that – We had Maxim. I'm sure you guys know Maxim from Temporal. One of the original authors of Cadence. He mentioned like if you're going to get a scooter, like you're going to rent a scooter from Uber, you start that scooter ride, and that scooter ride may last a little bit, like it's going to last for a while. And so what does that mean for the backend system? How do you model that scooter ride? You don't know how long the user is going to be taking the scooter ride for. There're different things that happen along the way. Maybe you stop the scooter for a little bit. Maybe you – I don't know. Press a button on the

scooter and it like gives you a snack or something in the future. Maybe there's a payment API that needs to be reached.

But in many case, it's kind of this long-lived workflows that are sort of like user sessions. But since this is kind of a new concept – I mean, it's not a new concept, but it's like – I think in the past, there is this something called the saga pattern that I think people used to use before systems like Cadence that sort of productionized and formalized how these workflows should work. Can you just give a little bit of an explanation for why this kind of workflow manager is a new piece of technology and why it's so useful?

[00:15:15] DG: Yeah, totally. I think one of the most interesting things about Cadence architecturally is that there is very little cost of adding new workflows. So this means that it is realistic to have a piece of code that runs per user.

So back to the scooter example, you can have this long living piece of code that is constantly checking the state of the scooter and updating any sort of changes that happen with the scooter. And there is very little cost of it, especially when – And there's essentially zero cost when the scooter is doing nothing.

The really powerful part of that is you can have millions of these workflows running at once. And in terms of the infrastructure needed, it really scales towards the usage. So if the scooter is doing nothing, there's no need for additional servers. It really allows you to have this model where you compare a piece of code to a user actually doing something in a way that it is truly pretty new.

[00:16:03] JM: Fouad, anything you want to add?

[00:16:05] FM: I think the key part for me is the product engineer who not only doesn't delve as much into the infrastructure side that defining the workflow sometimes can be a little bit cumbersome. You have to know exactly what you're doing and how to do it. That as we build our core product, aim to do, is have this kind of configuration layer where you can define policies. You can define the way you want the system to work. And then we then translate that into the cadence workflow. So kind of getting the best of both worlds where we are effectively exposing

the knobs to our customers to make the systems work the way exactly they want them to. But then have it implemented in this really fault-tolerant way, where there is no extra cron jobs that need to run. Everything is done in a declarative way.

[00:16:48] JM: Right. Explain that declarative workflow in more detail.

[00:16:53] DG: Yeah. Able to express exactly how you want – To be able to think of the workflow as like almost like an object. You're able to express different properties of it and have it so that you can actually write the code that expresses what those properties are at any given time. And by doing this, you get a predictability where you're able to basically account for every single state that the workflow could go into such that you can read test that actually is able to go through the pieces of that workflow and be able to test every single edge case in a way that is virtually identical to the way that production would treat it. And because of that, you get a high level of reliability, or least some proper testing.

[00:17:33] FM: So, when we're building an integration like with Slack, where normally you would have to rely on someone actually going in and testing end-to-end inside of Slack. We're able to mock-out what we would be receiving from Slack in terms of payloads and then have that all configured as part of our test cadence environment whenever we either test locally or we test in CIs. So we're able to ensure that these integrations that we build and maintain are actually continuously getting tested, even if they do rely on third-party code.

[00:18:00] JM: Okay. So let's take a step back. If I am building some kind of access control management, then I may want to use the SDKs that you guys have built, the software development kits. Or my want to use a REST API. Tell me about the developer experience for working with Indent.

[00:18:20] DG: All of our APIs are built around a technology called GRPC from Google, which GRPC uses protobufs to be able to in a binary format define exactly what the messages we receive. All of our serving infrastructure is built around GRPC. So the experience, you can either send a GRPC request or an HTTP request, and that, regardless, will be converted into the GRPC protobuf format.

And from there, we're able to use statically-defined types to be able to interpret the request. And that gives us a high ability to ensure that we're not receiving junk data, and that all information that we're receiving is what we expect.

[00:19:01] FM: And another half of it is actually focusing on getting up and running. So, providing these example web hooks where you can, for most of the services that you need to connect with, especially on the cloud provider side, you can get up and running and then also send examples for building a custom internal tool integration. So let's say you have your own custom RBAC table in your database and you want Indent to manage the approval flow, but then still have your web hook responsible for communicating with your database. That we provide the what we like to think about of as an escape patches, where whenever you want the system to work differently the way that you want it to be defined in your infrastructure, we can just have those escape patches.

[SPONSOR MESSAGE]

[00:19:44] JM: If you listen to this show, you are probably a software engineer or a data scientist. If you want to develop skills to build machine learning models, check out Springboard. Springboard is an online education program that gives you hands-on experience with creating and deploying machine learning models into production, and every student who goes through Springboard is paired with a mentor, a machine learning expert who gives that student one-on-one mentorship support over video.

The Springboard program offers a job guarantee in its career tracks, meaning that you do not have to pay until you secure a job in machine learning. If you're curious about transitioning into machine learning, go to softwareengineeringdaily.com/springboard. Listeners can get $500 in scholarship if they use the code AI Springboard. This scholarship is for 20 students who enroll by going to softwareengineeringdaily.com/springboard and enter the code AI springboard. It takes about 10 minutes to apply. It's free and it's awarded on a first-come first-served basis. If you're interested in transitioning into machine learning, go to softwareengineeringdaily.com/springboard.

Anyone who is interested and likes the idea of building and deploying machine learning models, deep learning models, you might like Springboard. Go to softwareengineeringdaily.com/springboard, and thank you to Springboard for being a sponsor.

[INTERVIEW CONTINUED]

[00:21:21] JM: The choice of GRPC, describe why you're building around GRPC. And I guess, by the way, the alternative would be like just raw JSON, right?

[00:21:31] DG: Exactly. The main reason that we were looking that we've looked into and decided to use it was that it provides an absolute structure to the way that information is sent and received by your application. And by doing that, you really can constrain the possibilities that your application needs to deal with. Additionally, because it is deterministic, we're able to generate client libraries across a variety of languages with pretty minimal effort, honestly. Most of the efforts around making sure that the user experience of using this API is good. But in terms of the actual transport layer, there's no additional work to be done other than in the generation, which takes seconds. So because of that, it gives us a great deal of flexibility, but also the ability to know that we can expect the type of information outcome into the system.

[00:22:20] JM: Right. With a protobuf, you are defining like typed systems of sending information from one endpoint to another.

[00:22:29] DG: Exactly.

[00:22:30] JM: So it's like a typed transport layer. In order to build an audit trail, there has to be some kind of log management. Where do you keep logs? What is the backend look like for that log management?

[00:22:43] DG: Yeah, totally. We're actually relatively flexible here. Currently, we largely have logs go out into a bucket. And then on top of that, we're using a tool that was originally developed by Facebook called Presto to do the analysis of those logs. And what Presto lets you do is across a wide variety of sources, it could be a local file system, it could be Postgres, Cassandra, or a bucket. You're able to write a SQL query that is able to actually go through

those raw files and give you result as you'd expect from a SQL query. And using that, we are actually able to have a pretty powerful experience of being able to request basically any sort of breakdown of the logs that we received.

[00:23:21] JM: So now we got two trendy infrastructure technologies, both Presto and Cadence. Is there anything else that the listeners can fill in their SEDaily trendy technology bingo stack with? I guess GRPC also fits the bill. Any other modern technologies you guys are using?

[00:23:43] DG: Yeah. So all of our infrastructure is orchestrated using Kubernetes, and it's crucial to everything we do. And that is configured using Terraform. And we use Flink to do the ingestion to complete your bingo board.

[00:23:55] JM: Yeah. It's enough to complete the bingo board, I think. You haven't said serverless, but it's in there somewhere.

[00:24:02] FM: Well, that's one of the ways. Yeah. You can't forget the Lambdas.

[00:24:07] JM: Right. Flink. You use Flink for doing what?

[00:24:13] DG: All of the audit logs coming in go through Flink and it's used to handle the ingestion. This is still in the early stages, but we are working to have it so that you can actually have real-time querying of any of the audit events that are coming in. Within seconds, you can be alerted of an audit event.

Then for the long term, we use Presto. So between the two, it gives a pretty good combination of being able to have very fast real-time capabilities as well as being able to have historical capabilities and being able to bridge the two.

[00:24:44] JM: Got you. For the Flink use case, all these logs are being generated by some of your middleware. Where are the logs coming off of and where are they going to?

[00:24:56] DG: Yeah, exactly. Any time that we receive an API request or do something on behalf of our users, so anytime there's a change, basically, we generate an event. And then that is ingested and then processed in a way that can be queried using both systems.

[00:25:10] FM: And then the other side of it could be an application that is defined by a customer that needs to send in audit events, or even a third-party application like the logs within Okta or the logs within G Suite where they each have their own discrete formats that we would be using Flink to process and normalize into our what we call standard format of actors, events and resources. So that way when we go to query it, we always have a reliable structure that we can query against.

[00:25:36] JM: Actors, events and resources. Say more about that.

[00:25:40] DG: Yeah. When you look at most audit log structures, they generally have some form of those three concepts, but generally under different names. Even if you look at something as simple as and similar as Okta's login logs and Google's, sign in with Google logs, they're actually completely different even though they have the same data.

So when you actually go into process or answer any kind of basic questions, the first step is data engineering. And one of the lessons I learned when I worked at Segment was you try to do the data engineering that's early in the pipeline as possible. So that way the people who need to consume it, the analysts, the security teams, the compliance teams, are able to rely on a normalized dataset. They can ask questions against.

[00:26:19] JM: So you define that normalization with Flink?

[00:26:24] FM: Yes. We have a protobuf that defines if you are sending data or when we're [inaudible 00:26:28] an event. But then when it comes to actually processing external, potentially third-party events, we try to, in our pipeline, normalize it.

[00:26:36] DG: Yeah. And one of the ways that we're able to do that relatively generically is using a tool that was developed by Google called JSonnet. And what Jsonnet lets you do is to find a deterministic translation and transformation of JSON. So what that lets you do is

essentially mutate any format into another in a way that is deterministic and should eventually complete. And because of that, it gives us a good amount of flexibility in terms of the types of templates we're able to accept.

[00:27:02] JM: So whenever something happens across your infrastructure, that generate some kind of event. That event is thrown to Flink for rapid processing. Do you have to buffer it? Do you have to put it into Kafka or something?

[00:27:16] DG: That will be the eventual plan once we – Right now we're putting it into RabbitMQ. But, eventually, we're planning on switching to Kafka. It's just a matter of pulling off and running Zookeeper as long as possible to be completely honest. That is definitely the goal. And probably within the few months, next few months, that change will happen.

[00:27:34] JM: Why don't you just use like Amazon Kafka or Confluent Kafka? Doesn't Confluent have like a serverless offering now?

[00:27:41] DG: Yeah. The main reason we have choose not to use cloud provider-specific services is we're really trying to be able to be portable across every environment possible. We have a keen focus on making sure that within the next six months we will be up and running on on-premise, and we already have some experimental things going, but we want to have a first class on-premise product very shortly. That's the main reason we've been trying to avoid vendor lock-in and cloud lock-in there.

[00:28:05] JM: Does confluent, or does Kafka have a Kubernetes operator yet?

[00:28:10] DG: It does. Unfortunately, for Zookeeper part is still in like the relatively early day. If you've got a running Zookeeper clusters, it's relatively trivial to get Kafka running on Kubernetes. And people have gotten Zookeeper running. But as far as I checked two months, there isn't anything – There is nothing that I felt comfortable operating yet. I totally believe it's possible. I think it will happen, and I think it just comes down to some of the annoyances of running Zookeeper.

[00:28:39] JM: Yeah. I think that the sales pitch of the operator pattern, does it makes the stuff running on Kubernetes that would normally have unexpected faults and where you need complex recovery logic, it makes that stuff simpler, which is obviously a very hard value proposition to fulfill, especially for a complex system like Kafka. It sounds like the technology is just not there yet.

[00:29:01] DG: Yeah. I think we'll definitely get there. I think it's just a matter of time when someone has the sufficient means and aims to solve that challenge. I think it will happen.

[00:29:11] JM: In the meantime, are you using RabbitMQ, which is just simpler but can be portable as well?

[00:29:20] DG: Exactly. Flink is able to relatively simply be able to switch from input sources like that. So we will eventually make the switch to Kafka. But once the data is actually within Flink, you do have some of the same properties in terms of durability as Kafka. It is basically just a transition. I'm hoping in the next few months that we'll be able to complete it.

[00:29:37] JM: There're really smart architectural decisions there. Why Flink instead of Spark?

[00:29:46] DG: The main thinking around Flink was that in terms of being able to take in sort of a fire hose and be able to ensure that every single piece of data is mapped correctly and is stored correctly, mostly just talking to friends who have operated both in that similar situation, it seems like the largest preference was with Flink. But we definitely did evaluate both.

[00:30:11] JM: That's interesting. So sort of off topic, but does that mean that – Because, I remember, first started doing shows on like the streaming technology and stuff like five years ago, four or five years ago, and you had Flink, Spark, Storm, Beam, blah-blah-blah. So many of these things. And it didn't seem that different, and now it seems like Spark has found more of a niche in just – You've got like a big data set the you want to do iterative processing on. And then you want to load something into memory and do successive processing jobs on it. Whereas Flink is kind of like aim your fire hose at Flink and just process things one by one in a streaming fashion.

[00:31:00] DG: Yeah. I mean, I think that's about right, where it's like architecturally, they're relatively similar. But I think the use cases have sort of diverged a bit, where I think that in terms of like the more machine learning and big data use cases, people have been moved towards Spark and largely use Spark. And because of that, I think the project is focused on that. And I think Flink sort of went another direction, which is sort of like the real-time processing where you're receiving a large amount of data constantly. And because of that, I think that the tooling has just sort of evolved along those sort of divergent mats.

[00:31:28] JM: While we're on the subject of engineering and these big distributed systems choices, anything that has been really, really hard for you guys to implement? What's been the hardest engineering problem?

[00:31:37] DG: I think that probably the hardest problem so far is ensuring that every single message is right exactly once. So, that is the main feature that we see getting out of Flink currently, is that just getting that guarantee. I think, fortunately, the tools have developed to the point where I think that a lot of these problems are traditionally hard. But it's actually simplified greatly. But I think that in terms of just adapting to these tools has been probably the hardest part and just understanding exactly how we want them to fulfill our use cases.

[00:32:09] JM: Can you say more about that?

[00:32:11] DG: Yeah, totally. If feel like it's kind of like instead of building something from scratch, the challenge is learning how to best use your tools. I think that that is a pretty big difference from at least a decade ago where a lot of these things just didn't exist in any sort of mature way, and building it yourself was the option. And now, it's I think more about just like how – What is your skillsets with these tools and are using these tools correctly? Are you using a hammer and a screwdriver at the right times and not trying to mix them together?

[00:32:41] JM: So it really has become a software architect's world.

[00:32:46] DG: Pretty much, yeah.

[00:32:47] JM: Did you make any mistakes early on in the architecture?

[00:32:52] DG: Oh, definitely. I think the first mistake that we're still mostly recovered from was writing than ingest – Trying to write that ingestion layer of buyer ourselves. We didn't immediately go to Flink. But it was quickly clear that a lot of these functionalities, it'd be foolish to not use, because otherwise we're just re-creating them and our version is likely going to be worse than the one that is used in production by the largest companies in software.

[00:33:23] JM: And the frontend, is there anything interesting there, or just kind of like React frontend with, I guess you, you could the Slack integration is kind of a frontend part. Tell me a bit about the frontend.

[00:33:37] FM: Yeah. Definitely, React, Typescript as much as we can. And then there is a fundamental piece around how we're building both the core infrastructure and the configuration that I was talking about earlier and how that bleeds into the frontend with this concept of blocks, which has been a growing trend in a lot of companies to enable their customers to have a certain level of configurability around how certain components work? How pieces of infrastructure get configured? Ultimately, that kind of WYSIWYG filter that you see in a lot of tools today, that our goal is to make sure that as much of the product, if we aim to be your access portal, can be configured just like the infrastructure can to work the way you want it to. So that's been a core foundational piece that, with this concept of blocks, allows us to define these data requirements. So let's say it's a query that needs to get run on Presto, and then the React components that then consume that query in a way that is closer to how you define Terraform modules and Terraform blocks. Then the traditional, surely, React style.

[00:34:39] JM: What's the deployment situation look like? You guys got all these big, chunky pieces and you're going to need to make it work for on-prem environments. You want to have as close to a one-click I've got RabbitMQ, and Flink, and whatever else, whatever other chunky pieces deployed. What does that look like?

[00:35:03] DG: Yeah. We've rested heavily on Kubernetes' abilities here. And we've tried to make it so that the deployments are as cloud-agnostic as possible such that it should be relatively trivial to be able to go to another cloud or on-premise as long as they were run in

Kubernetes cluster and deployed the stack. Because the vast majority is fully enclosed within the Kubernetes cluster.

[00:35:26] FM: Initially, we recognized, no one necessarily wants to get up and running by having run every piece of the infrastructure. So we run as much as we possibly can and then we separate the most security-sensitive bits. The thing that's actually accessing the secrets or the keys into their infrastructure, so it kind of give them the balance, that down the line, when they need it to run on-prem for specific, very sensitive workloads, then we could transition those workflows to run on their cloud.

[00:35:49] JM: Have you guys talked to the Replicated folks or the Gravitational people? These companies that help you get these things deployed more quickly on-prem?

[00:36:02] DG: Yeah, totally. Actually, the company that I first met Fouad was actually in a very similar space to them. So, very familiar of what they're doing. Yeah. No. I think that they do really good job at figuring out what are the requirements to go into these on-prem environments. And largely, how do you get Kubernetes working in those environments? I definitely see us sort of resting on other people's technical expertise there, because every environment is going to be so different, that I think that a great deal of flexibility is needed when you go into these on-premise situations. So we are definitely open to working with whoever is able to help us do that.

[00:36:38] FM: I think we're also trying to see a similar kind of initiatives from the cloud providers, like Anthos for GKE from the Google Cloud side.

[00:36:45] JM: But you're being kind of coy there. It sounds like you don't really know if you will need them or if you will be using them.

[00:36:53] DG: I mean – So out of cost sensitivity, I think that, ideally, we could do as much as ourselves. But I think that the reality is that that's likely not possible. And so, being able to be flexible here I think [inaudible 00:37:06] is being coy t o a point. But like part of it is just that is kind of the reality. It's just – Where it make sense, we will use them. But I think that just from keeping things in-house, that is definitely a goal where we can.

[00:37:06] JM: Well, it's also like your product is like your product is like by definition needs to be SOC 2 and whatever compliant and whatnot. So some of the value proposition of these companies where they help you get a distributed system that is compliant into production for on-prem environments, maybe that's just like de fact you've got it

[00:37:38] DG: We've always made sure to start with the compliant access and the complaint procedures even if we don't have the attestation yet, which we are now beginning to work towards for SOC 2. A key part of that of course is using Indent to, in our case, when we need access to cloud infrastructure resources. We don't just have product, even though [inaudible 00:37:55]. We don't just have product access by default. We have to actually request it for a period of time.

So, getting those practices and procedures in place early on. And then now that we're at a point where we have a reliable infrastructure, that we can operate in production. Actually getting that certified in all of our processes around are tested too.

[SPONSOR MESSAGE]

[00:38:21] JM: Today's show is sponsored by StrongDM. Managing your remote team as they work from home can be difficult. You might be managing a gazillion SSH keys and database passwords and Kubernetes certs. So meet StrongDM. Manage and audit access to servers, databases and Kubernetes clusters no matter where your employees are. With StrongDM, you can easily extend your identity provider to manage infrastructure access. Automate onboarding, off-boarding and moving people within roles. These are annoying problems. You can grant temporary access that automatically expires to your on-call teams. Admins get full auditability into anything anyone does. When they connect, what queries they run, what commands are typed? It's full visibility into everything. For SSH and RDP and Kubernetes, that means video replays. For databases, it's a single unified query log across all database management systems. StrongDM is used by companies like Hurst, Peloton, Betterment, Greenhouse and SoFi to manage access. It's more control and less hassle. StrongDM allows you to manage and audit remote access to infrastructure. Start your free 14-day trial today at strongdm.com/sedaily. That's strongdm.com/sedaily to start your free 14-day trial.

[INTERVIEW CONTINUED]

[00:39:50] JM: The process of getting compliance, is that tedious? What does that mean? Do you go to like a consultant, like a SOC 2 compliance consultant and then he says like, “Okay. If your data is at REST, it needs to be encrypted. If your data is in S3 buckets, it needs to be encrypted.” Stuff like that? What are these compliance decisions actually mean in practice?

[00:40:13] FM: Yeah. There are definitely a few vendors that will help you actually achieve the compliance. And then there's the actual auditor themselves, so just to make that distinction. And they can't be the same person for maybe obvious reasons. When you work with the vendors to actually prepare your policies, so that would be what you’re describing around how you're handling your data. What does access control look like? What is vendor management? Information security? All these other pieces. Business continuity? There's a set of policies that are expected by different compliance regulations that when you say that you are following a specific policy, the second leg of it when you actually bring your data, your evidence to the auditor, what they're looking through is for – In the case of SOC 2 type I, one days’ worth of data. I the case of SOC 2 type II, likely 3 to 6 months’ worth of data, to prove that you did not commit any violations. And that generally where companies go wrong, especially around access request, is people are Slacking, people are emailing them. There is generally a lot of missing data in terms of who is actually getting access that is not accounted for in their appropriate process.

[00:41:15] JM: How often do people actually get busted for these compliance policies? Does that actually happen?

[00:41:22] FM: Well, there this point of violations that will happen, and the other part of your policy is generally how you remediate those violations. And so there is generally some feedback loop where a company is expected to, every three months, do an internal audit, and that is part of your SOC 2 type II, is that you were doing these ongoing internal audits and taking steps to remediate it. So it's not intended to be this pure got you. But there are the standards as part of these regulations that say what does meet the standards for something like getting a SOC 2 type II attestation, that the auditors who are primarily accounting firms can only be as good as the data that they’re given. So there are requirements around you can’t tamper with any of the

audit evidence and things like that. But, again, they can only work off of as good as the data that they're given.

[00:42:10] JM: Let's talk a little bit more about the adaption and the usage. So maybe talk about a typical company that's using Indent and what their workflow is like for adapting it and putting it into practice.

[00:42:26] FM: Yeah. I'd say two main use cases, one on the engineering side that I'll start with first is the access to a cloud storage bucket. It's super common today for companies to have per customer buckets that they'll store data, whether it's audit logs or it's any other kind of metadata or even something that's operational. That a support Asian or an engineer might need temporary access to. And so in that case, we would have them deploy the web hook that runs in Google Cloud or runs in AWS to grant access to the buckets or to IM roles that give them access to those buckets. And then there's an expected approver. So that might be the head of DevOps. That might be a data owner for that given bucket.

Down the line, actually pulling contact set of things like Salesforce to understand who is the account manager or the CSM for that account. And then once we have those approvals grant the access and then keep them in the loop when their access is going to expire. So that the engineering use case.

And then the more broad use case, especially now that almost every company that solving this problem is using Okta or some similar identity provider is watching the watchers, so to speak, where a lot of people have admin access within those tools generally to add someone to a group or to look at the audit logs to a spot check something. But then that access doesn't get revoked. And so that's another common use cases. Managing admin access in systems, whether it'd be an identity provider like Okta, or it could be an internal tool. Or it's the kind of admin.company.com that people are particular concerned about.

[00:43:50] JM: What's been the process of winning over and onboarding customers?

[00:43:55] DG: Yeah. Generally, what we find ourselves competing against is some DIY solution either [inaudible 00:44:00] manual process, where let's say there's a form that goes into a sheet

or a Slack channel, and then there's the manual IT process that's centered around that. Generally, a lot of cat herding to figure out did the person ask for the right permission. Who is the right approver? Does this person actually need this permission?

And then there's the other side of it where companies that have a really high compliance burden or have a very sensitive internal security posture combined with enough engineering resources that they'll build that kind of admin or access.company.com. And those tools can only take on so much scope. They're generally but for very specific use case. And then as other teams in the company run into similar problems, they try to shoehorn their use case into that tool that generally breaks after a couple extra use cases. And then either they consider rebuilding it from scratch, and that's generally where we would be able to fit in and either augmenting that tool or replacing existing workflows that they've built on top of it. And then the other side of it is bringing in some new workflow that currently cannot be captured by that tool and Indent can solve. And then we expand from there.

[00:45:03] JM: What are the biggest sources of technical debt in the product? You could be honest.

[00:45:07] DG: Yeah. As you heard earlier, there is a little of a bingo going on for the infrastructure. So there's definitely a lot of operational overhead around just maintaining all of these different – The pipes under the hood. But what we recognized was the cost of me doing that maintenance. And we definitely plan to bring on and hire experts in the fields to solve around each of those problems and the kind of headaches that come out of them. But actually getting those pieces up and running. The opportunity cost of, let's say, building the new feature, which I've always very interested in kind of shipping something new on the product. But the balancing act of building reliable core infrastructure that people can use further or access control while still being able to move quickly as a regularly staged startup. I'd say that kind of operational debt that has some short-term impact that we're really excited to –We're starting to realize the value and definitely going to come into fruition in the coming months.

[00:46:01] JM: Basically, this is a job opportunity for anybody that's looking to be in SRE.

[00:46:07] DG: Yes, definitely. Absolutely. Infrastructure and product engineers.

[00:46:10] JM: Yes. If you're an SRE at Google right now, go to indent.com and find the email address and just email these guys. Let's see, what else? What haven't we covered? Anything interesting I'm missing at this point?

[00:46:23] FM: I think one interesting point to mention is around macro trends that are shifting. A couple years ago, as Dan mentioned, there weren't as many applications or there wasn't as much surface area that people might've needed access to. Maybe there is on-prem systems, especially when back when we're all working in offices, that you could trust people who are on the local office network. That now that people are working from home, there's more cloud applications. There's just more systems that you could potentially have access to that the problems of people getting access on day one, where you're just unable to do your job and you don't know what you need, because no one's actually told you. There hasn't been an update in terms of what you are granted off the bat. That that's really where we see Indent being a necessity in the modern IT stack, is you have these backends like Okta that's going to serve as the hub of where people are accessing using single sign-on, but then there's this change in management problem where it's unclear who actually should get that access and what the procedure should be. Where something that brings in the context out of these different systems and codifies these policies that generally live in a wiki document that say who should be able to request the access and who should be the appropriate approver and codifying all that so that companies can actively see and validate that the policies they've set are actually being met.

[00:47:42] JM: Right. Actually, this brings up something else. What's the big vision? Where are the adjacencies from where you currently stand?

[00:47:52] FM: Yeah. Our immediate focus is entirely around access, and the first piece of it is ensuring compliant access with organizations. And when we started this company, there is always this focus discussion the backdrop of Facebook in 2018 or the numerous incidents since that the stewards of our customer data are these tech companies and companies of all sorts that have an unbelievable burden to manage all of this data that when your primary incentive is to get people to have access to that data in the form of let's say ads or custom audiences, there's this competing misaligned incentives that there needs to be companies that are actively

working on behalf of people and ensuring that our data is only being accessed in an authorized manner.

And so when we look to the future and why we think tamperproof audit logs are so important requiring approvals, especially over time cryptographically, that we can search a branch out into enabling end-users to have control over their data. And you're starting to see this in drivers like GDPR or CCPA and companies that are spawning around it to handle data subject requests, which are amazing in helping companies facilitate the process. Working with them to ensure at an infrastructure and a at a system level, that those requests and those end-user requirements are being met and fulfilled. So when you request that your data be deleted or that no one is able to view your data unless they're authorized, that we can actually go in and enforce and validate that.

[00:49:23] JM: That sounds pretty bit. It sounds like a pretty big business.

[00:49:25] FM: Glad to hear it.

[00:49:27] JM: Let's see. Basically, you're saying the business of if I run a GDPR-compliant version of Software Engineering Daily, or softwaredaily.com, where we have a login system, somebody wants to delete their account and delete every comment they've made. Let's say GDPR requires that. You would be able to have indisputable trail of, "Yeah, that got deleted."

[00:49:55] FM: Exactly. Even if you're using a different system for actually submitting and handling and processing the requests – Segment and other companies have some sort of privacy portal that is intended to accomplish this. The next level is all of the internal systems, all the apps and tools that you might build internally, that would also have access. That mapping out where people could have potentially accessed the data and verifying that it actually is not getting accessed anymore is really important.

[00:50:21] JM: How big do you think that is? How big of a business do you think that is?

[00:50:26] FM: Well, I think everyone has their data living in so many systems and services that it's impossible to count. And that doesn't even start on the second party transactional data, that I

think this the problem that everyone deals with everywhere, whether it's us as end-users having our data floating around in the ether where the companies were expected to be the stewards of that data don't necessarily know where it is and who is accessing it.

We think that even just solving the company side of it, there's a massive business. And then when we start to actually tackle the privacy components [inaudible 00:51:00] for these other objectives that these companies also have, every company is trying to build trust with their customers. That Indent is intended to help companies build more reliable and trustworthy system.

[00:51:14] JM: Cool. Any future architectural decisions that you'd like to make? Let's say I give you five engineers today. Where do you dispatch those five engineers to doing?

[00:51:26] DG: Architecturally, I think we are relatively in a stable place. The main piece that I think that we are working on is just making sure that we can address more and more use cases. That noticeably is in like figuring out how we can make our model work for the multiple different types of systems that are processing this information and making it work in a way that is intuitive for a user and works with their workflow.

[00:51:53] JM: Okay. Well, let's see. Any SaaS products or platform as a service products, technologies that you've been mind-blown in terms of how useful they are? Underrated ones, particularly. You guys got boring technologies. That's a good sign.

[00:52:11] FM: Yeah. Especially when it comes to SaaS, we try to go with whatever everyone else is. That's the Saleforces and anything else that is expected to be industry-standard. We try not to reinvent the wheel on the app side, where we really need to be focused on the infrastructure and ensuring that we have the right capabilities in place for our customers.

[00:52:29] JM: Cool. Well, Fouad and Dan, guys, thanks for coming on the show. It's really interesting talking to you, and I think Indent is really impressive. It looks like you guys got a really good chance.

[00:52:37] FM: Awesome. Thank you so much. Thanks for having us.

[00:52:39] DG: Yeah, thank you.

[END OF INTERVIEW]

[00:52:50] JM: Join GitLab on August 26th, 2020 for GitLab Virtual Commit. It's an immersive 24-hour day of practical DevOps strategies shared by developers, operations professionals, engineers, managers and leaders. The attendees will hear from the US Air Force and Army. The GNOME foundation, State Farm, Northwestern Mutual, Google, more companies, many more companies about problems solved, cultures changed, release times that have been reduced. You can come and be part of the community that is just as passionate as you are about DevOps. You can register today at softwareengineeringdaily.com/gitlabcommit. Register for GitLab Virtual Commit by going to softwareengineeringdaily.com/gitlabcommit. Thanks for listening and thank you to GitLab.

[END]