Krish (products.snowpal.com) (00:00.839)
Hey folks, welcome to Snowpal Polyglot Software Development Podcast.

Our guest today is Jake Moshenko. Jake has been an innovator in the cloud-native ecosystem for over 15 years. After engineering roles at Amazon and Google, Jake founded Quay, the first private Docker registry, which was acquired by CoreOS. Jake then became an engineering leader at CoreOS, which was acquired by Red Hat and then IBM. He's now the co-founder and CEO of AuthZed, the company commercializing SpiceDB. Jake, thanks for taking the time.

Jake Moshenko (00:31.682)
Thanks for having me.

Krish (products.snowpal.com) (00:33.423)
We can kick this conversation off right away with, you know, just learning a little bit about your product, odds it, and then we can let that roll and have a bunch of questions around permissioning and permissioning databases. So it's an hour of learning for me. It's going to be a learning session for me personally. So a little bit about yourself, even though I've done a brief introduction, followed by some introduction to your current product would be awesome.

Jake Moshenko (01:04.694)
Yeah, I think in order to understand the story of AuthZED and why we thought it was important and why we decided to build it, we should sort of double click on some of the things in my past. So when we did Quay, which was our first company, when I say we, Joey and I were the co-founders of Quay and now Joey is my second time co-founder and our current CTO. So when Joey and I set off to build Quay, which was the first private Docker registry,

giving people private secure access to their Docker container images was one of the core fundamental value propositions of the product. And it was something that we didn't really have a good way of doing at scale. And it was sort of an afterthought. So we built the product. We built this private Docker registry. We had some idea that the permissions were going to kind of look like GitHub permissions. And we went ahead and we built that the way we imagined, using GitHub permissions.

And our very first feature request was, how can I do organizations, right? How can I group these things in a higher level way? How can I federate permissions based off of the teams and the groups that exist within my organization? We kept getting these requests, they just kept coming. It was teams and teams that were parts of other teams and things like that. And at the time we didn't really have a good scalable, flexible way of solving this problem.

So we did what every good engineer does, and we just started coding, just kept coding and coding and coding. And every time a new request would come in, it would be back to the drawing board. So this was kind of like our first experience with the problem of how do you do permissions flexibly at scale? And then as time goes on, as we were part of the CoreOS team and then part of the Red Hat team, and then eventually Red Hat got acquired by IBM, we just kept running into this problem over and over again. How do we do permissions flexibly at scale?

And we never really solved it. And then in 2019, Google published their Zanzibar paper. And the Zanzibar paper gives you a blueprint for how to do this. And I was reading the paper myself. I always like to take a moment to read Google papers, Google's white papers, because they're just excellent. And about halfway through the paper, I turned to Joey and I said, this solves the problem. This is a blueprint for how to solve the problem. We should go and we should build a service to give this technology to the rest of the world that doesn't work at Google.

Jake Moshenko (03:27.434)
and that isn't integrating against Google products. And that became the thesis for Authstead, and that's kind of what we've been doing ever since.

Krish (products.snowpal.com) (03:35.451)
Lovely. So to bring the problem closer, who's the target market? When should someone consider using AuthZid? What is the problem that they're looking to solve? And if they did not use AuthZid, how would they typically be doing it differently slash inefficiently possibly? Can we talk, if you could talk a little bit about that.

Jake Moshenko (03:59.594)
Yeah, we sell to products or we sell to companies with products in all verticals of all sizes. So there isn't really like any one ideal customer, right? Like the hallmark for when you might need something like what we provide is when you're trying to add a complicated or just unexpected permission or authorization workflow to your application and you're finding that you're limited by what you've chosen so far. So...

overwhelmingly what people are doing before they bring us in is they're just doing what I said. They're writing code and then they're writing more code and more code and more code. And this code ends up being pretty tricky, right? Like this is the code that guards the keys to the kingdom. If somebody breaches or gets access to something that they shouldn't have access to, that's usually like, you know, front page news headline. And so as you're changing this code over and over again, your product velocity is at odds with your overall security posture.

And so that's when it really makes sense to bring in a specialized service like ours to give you higher confidence, higher conviction that what you're building is good and it's not going to accidentally open up a security breach and also unlock your product velocity. Let you flexibly change and adapt to changing product requirements as they come in order to keep your software or your service keeping up with customer demand. Does that make sense?

Krish (products.snowpal.com) (05:25.747)
Okay.

No, it does. So let me take a simple example for me to digest this. Because at Snowpal, we are a product company as well. We have many products, like I mentioned quickly to you before the call. One of it is APIs and backends as services. So some of what you're telling me resonates quite a bit, because we might be solving some similar problems as well, in some ways at least. And we'll get to it at some point. But let me take a problem. If I were a mobile app developer or a web

developer.

or somebody who's building microservices on the server side. Just at the high 30,000 foot level, my high level question is, can, if I needed support to implement any form of permissioning, authorization, let me just say authorization. For any of these three types of quote unquote clients, meaning a mobile client, a native mobile app, a web application, or something that's entirely on the server side, either this is a microservice that's public, meaning other people are possibly using and integrating this.

or it could be in the VPC, meaning it's internal to the organization. If any of these people or applications per se had a need for permissioning, would odds Z be one of the things that they should potentially look at?

Jake Moshenko (06:46.85)
Definitely for the backend and the web application, the backend for the web application, that's kind of where we sit. So imagine that you get a request in, it comes in through one of your APIs into your backend. And it's saying that I've got a user and the user is identified by some cookie or some token or something like that. And the user is trying to do X, whatever that X is, right? Like delete an object, open an object, share something further, right?

Being able to answer the question, is this user who I have identified through some other means, right? Usually OIDC or some kind of identity provider. Is this person allowed to do this thing? We can help you answer that question.

Krish (products.snowpal.com) (07:29.167)
Okay. So let, you know what, just to get an idea as to where this sits, you know, maybe I should just, I don't draw very well, but I still think pictorially. Do you think I can just do a screen share so we can understand sort of where this fits into the picture then I can go with further questions? Does that make, is that all right? Okay. Let's try. We'll see how it goes. Yep. Absolutely. Yep. Let me share.

Jake Moshenko (07:47.198)
It's alright with me. This is a first, so let's trailblaze together.

Krish (products.snowpal.com) (08:02.635)
Okay, tell me if you see my blank Chrome screen. Okay, okay, I'm gonna... Okay, let's just start here. Just gonna put...

Jake Moshenko (08:08.118)
Yep, I do.

Krish (products.snowpal.com) (08:20.879)
I usually start, you know, I only understand three or four shapes. You're going to have a diagram that just squares and rectangles and, and I don't know, a circle. Absolutely. But just, you know, visually, sometimes it helps to understand this. Before I, okay, I took three of them. Let me draw. Let me take, I'm just going to pick another random shape here. I'm going to say

Jake Moshenko (08:27.531)
And arrows, yep.

Krish (products.snowpal.com) (08:59.295)
I'm just calling out for three categories. So for folks again, watching or listening to this, I mean, our podcast can, you know, there are people who watch it and listen to it, but when we do the screen share, you are better off actually watching it, just letting folks know. So this is back to my quest and let me pull this window over so I can see you at the same time. The screen share only challenges I have to make sure I sometimes don't see the guest when I'm actually doing the drawing here. Okay.

I'm building a mobile app, just a simple example. And I'm gonna skip authentication for a second. Let's say I use OAuth, maybe Google, Facebook, doesn't matter. I have integrated a third party social media sign in, I've logged in, so I know that hey, Jake is a user who's signed up or registered in my system. But what Jake can or cannot do has not yet been defined. I've not implemented anything. If I need it, if that is my ask,

Because let's say even at Snowpal, we have a mobile app. It's a project management application. And there's a number of features. And from an authorization standpoint, what you can or cannot do depends on a variety of things. So my first question, Jake, is if I had such a

need to satisfy, can AuthZed help me and save me time and not have me write more code than I would have had to write otherwise?

Jake Moshenko (10:24.266)
Yeah, in order to answer that, I need to ask you a few more questions about the mobile app. So the mobile app, when you're making a change to one of your projects, is that calling a backend in order to make that change?

Krish (products.snowpal.com) (10:28.107)
Sure.

Krish (products.snowpal.com) (10:35.163)
It is, yeah, it essentially integrates, maybe we're gonna have another diagram here. It does a number of things locally from a presentational standpoint. Obviously we do a lot of caching and stuff so we don't make requests to the server unless we have to, but otherwise everything is driven by N number of, at this point, rest, but soon enough, graph requests too. So you could take a combination of those two kinds of requests that are going to the server to get all kinds of information, right, from permissioning to all the way through creating content.

Jake Moshenko (11:01.238)
Yep, so I think if you add another square onto this diagram or rectangle or whatever, and we call that the back end or a trapezoid.

Krish (products.snowpal.com) (11:04.732)
Okay?

Krish (products.snowpal.com) (11:14.089)
Okay, let, actually, yeah. Okay, so this diagram, should I just put it here, right? I'm just gonna say, I'm gonna say snow pal back end. Okay, I'm gonna go from here. And I presume this is gonna be the same for all of these, so I'm just gonna connect these arrows here. Yeah, I would.

Jake Moshenko (11:32.318)
Yeah, although microservice could sit behind the backend, right?

Krish (products.snowpal.com) (11:35.943)
Yep, I was just gonna, okay, let me leave that arrow for now. I don't know why the arrows don't go all the way, maybe because it's a rhombus. Okay, so there is this backend, and now should I just connect this to odds-in? Okay, so now let's go with the flow here. So the user signs in, let's pretend that they didn't and Apple sign in or something of that nature. I'm just gonna write it here. User signs in, yeah.

Jake Moshenko (11:49.183)
Yep, that's exactly it.

Jake Moshenko (12:01.154)
Sure.

Krish (products.snowpal.com) (12:09.791)
Right, so it's odd, odd to sign in. And then they come here. Now Jake is valid in the system, in Snowpal system. Now there is many facets to the backend. So I mentioned a project management app. So let's say if I took this, I'm just gonna say amongst many things that this does, let me actually pick another shape here. Okay, I'm just gonna just make it two circles. And I'm gonna call one.

Krish (products.snowpal.com) (12:44.743)
all other functionality

Krish (products.snowpal.com) (12:55.899)
And then this is what I'm going to say permissioning.

Krish (products.snowpal.com) (13:07.06)
Okay, so I'm just trivializing this to say, actually let me switch the order of these circles.

Krish (products.snowpal.com) (13:32.827)
Just gonna give it a color here.

Krish (products.snowpal.com) (13:39.231)
Okay.

What I mean by this Jake here is this, whatever the product does, right? We have a project management app, for instance, it has everything to do with project management. That's at least one of our products. So I'm just gonna say all of the functionality that obviously we would have to implement as a provider, as an implementer of that application. But what we don't have to do is permissioning, which obviously we have because we have our own permissioning APIs. But for somebody else who's doing it, for instance, am I correct in saying that, hey, you know what, build your solution,

on your core customer problems, implement all of your functionality that's specific to your domain, your customer problems. But when it comes to permissioning, don't write any code, leverage something like oddset. Would I be correct if I started the conversation there?

Jake Moshenko (14:25.674)
Yeah, that's an interesting way to think about it. So what your code normally has, and you can correct me if I'm wrong, but code in a backend usually has something like, if the user has access to this, right? And usually the, do they have access to this, is either a function call or some really complicated if conditional or something like that. Then allow that thing to happen. Otherwise return like a 403 error, right?

Krish (products.snowpal.com) (14:51.871)
Correct, right?

Jake Moshenko (14:52.574)
And so what AuthZED does is it gives you a common way to talk about that really complicated condition of whether somebody has access to do that. And if you think about it from that perspective, when you say like, you know, if the person is an admin, they're allowed to do this, right. And that's, that's like level zero, right? Like that's what you imagined that the code should look like when you first set out to write it. But what ends up actually being the case is what it means to be an admin gets more and more complicated.

And who is allowed to take that action sometimes bleeds into other roles, right? Like you can have super admins and you can have billing admins and you can have writers and editors and managers, and you start adding all of these concepts, but you really, really still want to be able to ask in that one if statement, is this person allowed to do this thing? And AuthZED gives you a web service to talk about and to collaborate on how to do permissions and we will go ahead and we will do all the computation to figure that out for you.

Krish (products.snowpal.com) (15:39.551)
Correct.

Jake Moshenko (15:49.83)
And on top of that, when you talked about microservices earlier, those microservices can now also ask that question without having to have access to that code inside the backend itself. Does that make sense?

Krish (products.snowpal.com) (16:01.423)
It does make sense. So what I'll do is I'll move microservice sort of here. Now the microservice, I'm gonna draw a couple of different arrows and we can make the adjustments here. Because again, a lot of these arrows and the direction really depends on the specific, the business and the use cases. But the reason tell me if this arrow, the way I'm drawing it here. So if I have a microservice, it does a certain, let's say I'm a third party, I'm neither Snowpal nor AuthZed.

that's actually building a service for either my own product, my own company, or I'm building something that's made available to other users outside my, in the DMZ, for instance. So when I build this microservice, I have a certain piece of functionality that I'm implementing. Let's say I'm building a microservice for the food industry. I have some functionality that's around delivery and food ordering and whatnot.

I can use my own code, my own functionality to support it. Or let's say you use something like Snowpal, which is another company that provides APIs to implement that functionality. But when it comes to permissioning, I don't have to implement permissioning because I'm actually gonna say, you know what? The part that deals with permissioning, specifically let me not implement it or not even worry about it. I'm gonna leverage AuthZ and its managed service to support all of my,

requirements. Does that make sense? Okay.

Jake Moshenko (17:25.91)
Yeah, absolutely. And this is a really common story for us. We do this stuff all the time.

Krish (products.snowpal.com) (17:31.727)
Okay, so which means if people didn't use that, I'll talk about something we do, which is actually very similar, but there could be some, you know, synergy here.

The idea here then, Jake, is people shouldn't spend time worrying about designing a schema for authorization, the complexities of what it is, the flexibility, the scalability, and all of those items. Because these are the idea that it's a generic enough problem when it comes to permissioning, that you don't want to reinvent the wheel as a provider, as a service provider, either for yourself or for your clients. So you basically have the time back

given back to you and the money, so you can focus on your core customer problems and anything that deals with any form of permissioning and authorization and security, you're like, you know what, let me not worry about designing, implementing, maintaining, testing, managing, deploying and yada, yada. Let me just connect to odds and just leverage it as a managed service. Is that the core premise?

Jake Moshenko (18:34.122)
Very close. The only thing that I'll clarify is that most applications require some kind of custom permissioning. Like if you were trying to adopt an out of the box permissioning schema, it might make an assumption that you only have one resource or you only have one user type. With AuthZ, you actually get to model the permissions in our schema language exactly how your app works. So you can get 100% coverage over all of the crazy sharing and fine grain authorization and, you know,

tiny piece by tiny piece federation of access to data. You can model all of that in our schema language and we'll just give you a common API that you can talk about those things and we'll handle all of the computation and we'll store all of the data for you and we'll handle making those decisions and you just have one API that you can call that says, am I allowed to do this? And it'll say yes or no.

Krish (products.snowpal.com) (19:25.291)
Okay, makes complete sense. So let me, I'm gonna share my entire screen because I, do you see my screen?

Jake Moshenko (19:31.562)
I do, yep.

Krish (products.snowpal.com) (19:32.571)
Okay, I think this makes sense. Let's take a more specific example so we can dig a little bit deeper. Typically, again, for folks watching a lot of our collaborative podcast center on a topic, not so much a product, but again, it's not predefined, it's not scripted. We just let it go as the conversation flows. In this scenario, to me personally, I was interested in OZ and its permissioning models for multiple reasons. One, also because we have APIs that solve similar problems. I wanna see what commonalities

and what synergies there possibly could be. So we just exploring the product to begin with. And then the idea is not necessarily to pitch any of the products. It's not the idea behind the podcast. It's just to educate folks and make people aware. So you can, you know, folks can do their due diligence. Just a high level disclaimer there. Okay, I'm gonna take, I'm gonna go, you see this screen as well, right Jay? Okay, I'm just gonna log in. It's, it's.

This is the production. One of our, this is our web app. It's the production version. So I might switch to a local version, but let's say, you know what, I have some data here that I might not be able to share, sorry. So let me just go to a local version and pick up some dev data. Okay, let's take an example. This is a web application.

that let's say if I go to my main page, one of the main pages, this has a list of keys. You can look at this as a project, a list of projects for instance, right? A project or a space or a key is our terminology. You create content and this is your highest level container. Now, when I create this container, I go into a container and I break my problem down into different pieces. Here I say taxes, the tax season is coming up. I said 1040 and you might have other forms to file like say 1120 as if you're a business and stuff.

like that. This is content I create. Now I need to share. I want to be able to collaborate. These are all dev users. That's why you have these fake names. Currently, it's using R implementation. If I used, hypothetically, if somebody is building a system like this and they needed to implement permissioning, I want to be able to share this block, this resource, with Parag with two users. And I grant them different levels of access, read, write, and admin. This is a user interface.

Krish (products.snowpal.com) (21:49.621)
that we're looking at, but there's a rich backend system that supports this. Could we do something like this possibly using odd set without having to design any of this J?

Jake Moshenko (22:00.414)
You would still create your own UI elements, but we would handle all of the storing of the data. So when you shared a block in your terminology with another user, you would store that in

AuthZED. You would say, Parag now has admin access to this key, I believe is what you called, or block.

Krish (products.snowpal.com) (22:12.176)
Great.

Krish (products.snowpal.com) (22:19.847)
Okay, so let's store it here, right? I'm gonna say, I'm gonna increase this font here. So we are saying store, sorry, say that again. I'm just gonna say the request from Snowpal goes to odd zed. The idea here is to be able to share this piece of content which is a block with Parag as a user. What would be my experience from an integration standpoint? Okay. Is it a REST API?

Jake Moshenko (22:41.898)
Yep, you would call an API. It's called Right Relationships. It's gRPC, but we do have a rest gateway.

Krish (products.snowpal.com) (22:50.365)
Okay perfect okay I'm just gonna say call API okay.

Jake Moshenko (22:53.45)
Yep, and it would be called Right Relationships.

Krish (products.snowpal.com) (22:56.203)
call API to write relationships, okay.

Jake Moshenko (23:01.546)
Yep, and what you would do is you would tell AuthSED that Parag is now whatever access you wanted to grant, I forget from the previous screen, like let's say it was read access.

Krish (products.snowpal.com) (23:12.743)
I'm just going to say, let's say read access, right? In other words, this is my intent. Let me actually place my, put my intent here. Sometimes, okay. My intent is to grant read access on resource, what is it? 1040, two, two.

Jake Moshenko (23:34.638)
I think it was like 1120, yeah, 1040.

Krish (products.snowpal.com) (23:39.822)
Let's say this is my intent. This is what I'm intending to do. Right now we'll say, okay, the first step is call API, write relationships, okay?

Jake Moshenko (23:48.914)
And in the body of that request, you would say that Parag is now a reader of 1040. That's what you would tell us.

Krish (products.snowpal.com) (23:57.991)
Okay, in the body of the request. Okay, say that again, Jay.

Jake Moshenko (24:03.19)
Parag is now a reader of 1040.

Krish (products.snowpal.com) (24:10.623)
But before I go there, Mike, sorry, I'm just gonna interrupt you just to follow along here.

Jake Moshenko (24:15.532)
Yep.

Krish (products.snowpal.com) (24:16.755)
This is a particular type of access. Like read, in our case, we have, for example, in our web ACL model, we have read, write and admin. Let's keep it very simple, right? It's, let's say not teams, individual members, let's forget roles, let's talk only privileges. There are three very granular privileges. One is read, one is write and one is admin. How do I register this fact with odds-z? Because I have to tell AuthZed, hey, for Snowpal backend, these are the three permissions

are available to be granted to a resource because if I did not do that, if someone tries to grant some other privilege, which is neither read, write or admin on that resource, odds that it's gonna have to fail because that's not something it recognizes as a pre-registered list of privileges for this particular system because that could be a different system that has a different set of privileges and dependencies on odds that. Let me know if my question's unclear, but if that makes sense, how would we go about it?

Jake Moshenko (25:11.966)
Absolutely. No. Yeah, can I get you to open up another tab?

Krish (products.snowpal.com) (25:18.771)
Yep, I have odds that open here if you want me to go there.

Jake Moshenko (25:21.923)
No, just open play.authzed.com, which is our playground.

Krish (products.snowpal.com) (25:25.963)
Play.odds.z.

Krish (products.snowpal.com) (25:31.147)
Okay.

Jake Moshenko (25:31.886)
OK. And if you want to select example schema.

Jake Moshenko (25:39.462)
I think this might actually be the one that I would pick. It's a little small. But yeah, do simple role-based access. Just click that to make sure we're starting from a known place. Yeah, so you could imagine this is a schema. So remember earlier when I was telling you that every application has its own way that it thinks about permissions? So in this case, we're protecting a resource called a document. But in your case, that resource would be called a block.

Krish (products.snowpal.com) (25:46.795)
Okay, simple rule. Okay.

Krish (products.snowpal.com) (25:59.56)
Right?

Krish (products.snowpal.com) (26:07.535)
Okay, so in other words, the first thing I would do is when I make the request to odd z, I'm just mapping the block to a document because you obviously don't understand block, but you understand documents, I need to do the mapping.

Jake Moshenko (26:17.854)
No, no, no. Nope, not at all. Just change the word document to block.

Krish (products.snowpal.com) (26:24.516)
Oh wow, okay, like this.

Jake Moshenko (26:26.142)
Yep, so now we're starting to tailor this pre-existing schema directly to your app.

Krish (products.snowpal.com) (26:33.135)
I see what you mean. So in other words, this is, so let me, my first question here. And again, this is a couple of engineers talking folks who are going to go back and forth. This is like a design session, if you will, right? A typical of our podcast, but this is fun as well.

Let's say if I, so this is a DSL. The first question I have you're looking at this now is, is there an odd Z domain specific language that we have to quickly sort of understand?

Jake Moshenko (27:03.318)
You're looking at it. And most of the functionality here is spelled out in these comments. So what you have are you have definitions. Those are different object types that might exist in your system. Just based on what I saw earlier, your object types would be block and key at the very least.

Krish (products.snowpal.com) (27:14.315)
Okay. I'm gonna.

Krish (products.snowpal.com) (27:21.835)
say DSL, right? The first thing is, okay, I'm just gonna, just to make note of the items here. Okay, sorry, continue, yep.

Jake Moshenko (27:30.715)
Yep, yep, so it sounds like your permission system would have block and key. So.

Krish (products.snowpal.com) (27:35.579)
It would have block, not key, just say block. And we have another item called pods. For instance, if I go here, pods can be at the same level as blocks. I can hierarchically go below blocks and I have a pod and I can create a pod saying, I don't know, form, let's say stocks right under 1040. So this is breaking it down. I can either grant access to pods independently of the pattern that it's sitting on, or I could grant access to the pod and say everything,

under my structure will also be granted access. So we have a few different ways of doing it, but let's say blocks and pods are the two items. So block is one and pod is another.

Jake Moshenko (28:15.41)
Okay, that's perfect. So let's do block first, and then we'll copy paste it for pod, because I think they're going to have very similar models. So under block, you see where it says relation reader and relation writer. Those are, you can think of those as the roles, but those are the relationship that a user can have with a block.

Krish (products.snowpal.com) (28:22.812)
Okay?

Krish (products.snowpal.com) (28:27.645)
Yep.

Krish (products.snowpal.com) (28:33.951)
So this writer, is that something I can change or is that the term that is that what odds it is expecting? Okay, so I'm gonna say writer relation, read relation, write something like this, right?

Jake Moshenko (28:41.246)
You can make it say whatever you want.

Jake Moshenko (28:47.402)
Well, hold on a second, because if you look down below, there's permissions, and those are separate. So permissions are the computations that you're going to do over the relationships.

Krish (products.snowpal.com) (28:57.351)
Okay, so should we add the permissions first for like read, write and admin?

Jake Moshenko (29:02.078)
Well, what are, so the permissions represent an API for what a person is going to try to do to a block. So what can you do to a block?

Krish (products.snowpal.com) (29:08.86)
Okay.

you can from a collaborative collaboration standpoint, from a security standpoint, or from an authorization standpoint, this is what you can do to a block. You can go to a block, and you can actually say, I want to grant access. You first look up a list of users, and then you pick the user you want to grant access to. And you can grant one of three kinds of access, read, write, or admin. Let's say that's the simplest form of the requirement.

Jake Moshenko (29:25.541)
Mm-hmm.

Jake Moshenko (29:37.324)
Mm-hmm.

Krish (products.snowpal.com) (29:39.723)
Does that answer your question, June?

Jake Moshenko (29:42.002)
Yeah, so there's read, write, and admin. Do you actually want me to model it? Like I could share my screen. Is that a? Yeah. Let me open up a tab then.

Krish (products.snowpal.com) (29:47.739)
Yeah, we can... Yeah, absolutely.

Jake Moshenko (29:57.534)
I've just got a ton of experience modeling these things.

Krish (products.snowpal.com) (29:59.664)
No, no, absolutely.

Jake Moshenko (30:02.75)
So I'll start from the same.

I'll start from the same DSL as soon as I get everything running.

Jake Moshenko (30:13.246)
So basic, simple role-based access. And you can do, by the way, you can do more than just role-based access. You can do all kinds of entire screen, screen two. All right, can you see my screen now? Okay, so I'm going to go and do block here. The specific things that a person can do to a block, we talked about you can grant access, right?

Krish (products.snowpal.com) (30:27.931)
Yep, I can.

Jake Moshenko (30:42.654)
And right now I'm just going to define these as nil. We're not going to say how they're computed. You can view them or read. Is that what you call it? What would you call the verb? What am I trying to do to a block? View it.

Krish (products.snowpal.com) (30:42.667)
All right.

Krish (products.snowpal.com) (30:46.475)
Okay.

Krish (products.snowpal.com) (30:53.227)
Three verbs, right? I guess it's a noun I wanna say, read, write, and admin. Those are the three privileges that you can grant on a resource to a user.

Jake Moshenko (31:03.166)
Okay.

Jake Moshenko (31:11.178)
Okay, so we call those permissions. Those are the specific things that you can do to a block. Now, how we figure out... Yep.

Krish (products.snowpal.com) (31:16.955)
Okay, so I have a question. Sorry, I have a question there. I get the read, write and admin part of the permission. Why is grant access also a permission?

Jake Moshenko (31:26.102)
Right, because you want to ask the question about the action that the user is trying to do. And I can explain why it matters, the distinction between the role and the thing that they're trying to do in a second, because that ties back into the flexibility that we talked about. Okay, now relations are how a user or another object can relate to that object. So here we're saying a user can be a writer of a block, or a user can be a reader of a block.

Krish (products.snowpal.com) (31:41.163)
Sure, okay, makes sense.

Jake Moshenko (31:55.862)
or a user can be an admin of a block. And now when we talk about the permissions themselves, so this would allow us to ask the question, is Jake allowed to grant access to block 123? Pretend that's some UUID. This is the question here, and this is the declaration.

Krish (products.snowpal.com) (32:00.933)
Okay.

Krish (products.snowpal.com) (32:20.337)
Okay.

Jake Moshenko (32:25.11)
This would be Jake is a writer of block 123. Does that make sense? So we're decoupling the idea of what relationship you have with the data to the actions that you can do on the data. And now.

Krish (products.snowpal.com) (32:40.683)
Okay, let me ask you this. When you say on the right side of 11, 12, and 13 of the colon, you have user. Is it, you're using user because on line four, you have user. If I change line four to users, would 11, 12, and 13 become users? Okay.

Jake Moshenko (32:58.186)
Yeah, you'd have to change it. We usually talk about things in the singular, but for sure you could, users is fine.

Krish (products.snowpal.com) (33:05.979)
No, no, I didn't. I just picked up. No, let's just go to Apple user. I think that's a better, I just want to change that. Let's say Apple user.

Jake Moshenko (33:12.652)
Yep.

Jake Moshenko (33:16.69)
Okay, so now we're saying that an Apple user can be a writer of a block, an Apple user can be a reader of a block, or an Apple user can be an admin. Now, when it comes to the specific things that you want to be able to do to a block, we come down here and we can say, who is allowed to grant access to a block? My guess is only the admin of the block is allowed to grant access. Who is allowed to read a block? My guess is a reader is allowed, and writers.

Krish (products.snowpal.com) (33:36.779)
Correct, yep.

Jake Moshenko (33:45.906)
are allowed and admins are also allowed to read the block. Is that correct? Yep, and then who is allowed to write? My guess here is it's writers and admins. And then for administer, I'm guessing only admins. Usually administer would be something like delete, right? Or rename, right? Only these people can do those, or only the admin can do those specific administrator level actions.

Krish (products.snowpal.com) (33:50.955)
Correct, correct, yep.

Krish (products.snowpal.com) (33:57.279)
Perfect.

Krish (products.snowpal.com) (34:14.335)
Correct, so the permission to me is a privilege per se, right? It's very granular. So read, write, delete, and rename. Is there a way, can I have a role that actually encompasses read, write, delete, and rename?

Jake Moshenko (34:29.986)
Read, write, delete, and rename.

Krish (products.snowpal.com) (34:32.615)
Like if I want to group all of those privileges or permissions into a role, so I grant access on that role to a user, would I do that? Is that a concept of a role?

Jake Moshenko (34:41.302)
Well, that would just be, these are the roles up here, right? These are the.

These are the things that you can do, like your direct relationship, the role that you have on a particular block.

Krish (products.snowpal.com) (34:53.871)
Okay, so let me read this again. So the permission, read as reader, writer, and admin. Write as writer and admin. Delete and rename can only be done by an admin. So you define who can, in other words, 16 through 20, they answer the question as to who can do what essentially. Is that right, Jake?

Jake Moshenko (35:13.246)
Yeah, and why, right? Why can I grant access? Well, because I'm an administrator on the block.

Krish (products.snowpal.com) (35:21.747)
Right, okay. And then I'm trying to map this to a traditional schema, whether it's SQL or NoSQL, doesn't matter. In my mind, I'm trying to see somebody who's used to either of those or both of those, how do I take this DSL and map it to that visually? So the permissions, I think I'm getting the... Okay.

Jake Moshenko (35:39.138)
So we handle storing all of the data. So what we do is we build a directed graph out of the facts that you tell us or the relationships as we call them. And so that's where this part comes into play. So I'm gonna get rid of these just cause they don't make sense in our current schema. But remember earlier when I said that you would tell me that Parag is a reader of block one, this is where you would do it, right? So here we would say that block.

Krish (products.snowpal.com) (36:01.279)
Right?

Jake Moshenko (36:08.21)
And we'll call it 123, right? And this could be any identifier. Usually it's a UUID or something in your database. And then here for the relation, we're going to say reader. And then we're going to say that Apple user Parag.

Krish (products.snowpal.com) (36:14.475)
Correct.

Jake Moshenko (36:25.196)
is a reader of Block 123.

Krish (products.snowpal.com) (36:27.835)
Okay, so two paths, right? So there are some design time actions and runtime. Can I say that there is design time and runtime and what we were looking at earlier was more design time? Okay.

Jake Moshenko (36:36.854)

This is design time. Yep. This is runtime. These are the kinds of, remember earlier when I said you would call write relationships and you would tell us that Parag is a reader of a block? That's what this does. So when you call write relationships, this data gets stored in AuthZed. And now anyone that has access to the AuthZed endpoint can answer those questions about what Parag is allowed to do to a particular block.

Krish (products.snowpal.com) (36:46.603)
Correct? Right?

Krish (products.snowpal.com) (36:53.037)
Okay.

Krish (products.snowpal.com) (37:01.131)
So in other words, the first thing I would do is from a design time standpoint

when Snowpal or any other system is integrating and we'll go to the previous diagram that I was drawing there to odds it, it's actually going to establish the design time before any user signs up, registers, does any of that stuff. You're like, hey, these are the rules of my system. Here you go. Can you accept the rules and register slash configure the rules in your system? And that's what we're doing through this schema here. And this schema would be sent to you

PC call as well, right? So we would register this through, but it just done. I guess we'll go and we can go to the details at a later point, but what I'm, in my mind, I'm imagining that this happens in a different piece of the code at a different point of time, because you're sort of doing the configuration. But once this is done, in other words, you do this before you go live on your long story short, you're gonna have to do this before you go live. Otherwise, you know, there is no runtime there. After you do this, when you go to test relationships, Jake, if you go to the second tab, you're showing this through a user interface,

Jake Moshenko (37:36.994)
Correct.

Krish (products.snowpal.com) (38:07.628)
this entry comes into play by an API endpoint call as well, by a gRPC call. I'm like, you know what? Take this block ID123, some GUID or UID, and this is a relation, this is the user, this is the user, et cetera. And if...

If I send you a relation that's not reader, you're basically going to reject it because the previous configuration from the schema standpoint established the fact that this system only supports a reader, a writer or an admin, right? So that call would fail because your checks would fail saying, you know what, this doesn't make sense. I reject this request for instance. And then this view that we're looking at here, is it...

Jake Moshenko (38:37.102)
Correct.

Jake Moshenko (38:41.226)
Yep, exactly.

Krish (products.snowpal.com) (38:46.535)
Is it sort of once I have these users in my system, if I want to quickly go in and look at it visually through an interface, you provide this sort of a design, not design, but sort of a tool, an IDE, or like a database client or whatnot, to be able to look at the list of users that we have essentially.

Jake Moshenko (39:04.886)
We give you a CLI to look at this data and being able to attach this playground to a live running system is coming soon. Looking at the raw data from a live production system isn't as useful as you would think because this is going to be like block UUID has reader of user UUID, right? So without being able to decode that and without being able to connect it and enrich it from the actual backend database or from the identity provider.

it's just going to look like a lot of opaque identifiers.

Krish (products.snowpal.com) (39:37.619)
But this is read-only. In other words, if I had reader here, let's say the actual request was made through an API to this gRPC call, but I can come back here. Let's say I'm in support or something. I'm like, you know what? Actually, Parag should have actually had admin and not read access or whatnot. I presume I can come in and edit and pull the dropdown and just switch the privileges out here.

Jake Moshenko (39:59.53)
You would do that again through an API call. So, and you can use our CLI or you can use your own, what usually happens is whoever makes the decisions about what access Parag should have to a given block, would just go back into your UI and then change it from there. And then that would make the API calls to actually change this to like writer.

Krish (products.snowpal.com) (40:02.603)
Okay.

Krish (products.snowpal.com) (40:16.337)
Okay.

Krish (products.snowpal.com) (40:23.73)
Okay.

Jake Moshenko (40:25.642)
Yep, so that's how that works. And then we also have like some ability to, let's say we have a check watch and let's say that we have the idea that Parag should be able to grant access to a block. And here this red X is telling us that Parag can't grant access to block one, two, three. Let's take a look at why. So only admins can grant access. And this is saying that Parag does not have admin on that block.

Krish (products.snowpal.com) (40:54.063)
Right, okay.

Jake Moshenko (40:55.242)
But if we go ahead and we give Paraga Admin.

then this will go true, right? And let's say we made a mistake in our schema and we decide that writers actually should be able to share, should be able to grant access. We can come over here and we can say that grant access is admin plus writer. And now it works because Parag is a writer, right? So this is what I was talking about, that flexibility. You can update this schema at any time, even on a live system.

Krish (products.snowpal.com) (41:01.563)
Perfect. Okay, makes sense.

Krish (products.snowpal.com) (41:23.087)

And then.

Jake Moshenko (41:29.554)
And changing these definitions for permission is zero cost.

Krish (products.snowpal.com) (41:34.719)
So when I make this change here through an API, for instance, I'm just trying to think, okay, so I had you, there was a certain different kind of access. What was the last change you made there? You added writer to grant access.

Jake Moshenko (41:47.594)
Yep. So if I get rid of that, you'll see that Parag loses access again.

Krish (products.snowpal.com) (41:53.149)
Okay, I see what you mean.

Jake Moshenko (41:54.815)
Is Parag someone on your team?

Krish (products.snowpal.com) (41:57.283)
No, actually not. You know, for the dev data, it only depends on when you actually create it. I think when I was running that locally, at least that instance of what is running, it was when I think Twitter had, or XS, it's now, had switched CEOs to a new CEO. And I believe his name was Bharat. So that was the day we created some of the data. And we're like, well, what name should we come up with? And then a bunch of our data created at that point of time.

Jake Moshenko (42:20.834)
Hahaha.

Krish (products.snowpal.com) (42:23.759)
ended up being products. So if we created one yesterday, it would be something else. If you create one in the last month, it would have been Jensen or something. People are in the news lately or something like that. This gives an idea. So let me go back to what I was, let me, if you stop sharing, yeah.

Jake Moshenko (42:40.686)
You want me to stop sharing?

Krish (products.snowpal.com) (42:42.931)
This is great. So I think that gives people the context for them to dig a little bit deeper. We showed the system so people can certainly go check out and reach out to you when they have questions. So that gives them introduction and, sorry, go ahead.

Jake Moshenko (42:54.042)
Usually, yeah, usually when I get people like yourself on the phone or on a call where we're modeling, they really get to the wow factor. When I go ahead and I add in like a higher level or a lower level object, and I tie it all together right in the schema, and I say that you have access to this because you're an admin on the block that it belongs to. Or blocks can belong to other blocks, which can belong to other blocks. And we have the recursive cascading.

Krish (products.snowpal.com) (43:17.138)
Right.

Jake Moshenko (43:23.858)

all the way up and down the hierarchy. And we add things like super administrators at the platform level, and they get access to absolutely everything. And that's when people kind of go, whoa, right? Like you don't think about those things as requirements ahead of time, but when you need them, it's usually more code, more code. I'm writing more code. And when you have something like AuthZ, you're just going in and you're changing the schema and you're extending the functionality seamlessly, immediately, and it just becomes a new capability for your app.

Krish (products.snowpal.com) (43:36.127)
Right.

Krish (products.snowpal.com) (43:51.055)
No, it makes a lot of sense, right? In the interest of completeness, I want to say that I couldn't agree more with the problem, right? The problem that you're solving. But in the interest of completeness, I want to share something for like two minutes. So please humor me for like two minutes here, just so I can say that we completely recognize that problem. So let me share my screen and go back here. OK, I'm going to go to.

Krish (products.snowpal.com) (44:22.059)
Do you see this postman page? I'll make sure I'm sharing it. Okay, cool.

So this is our version, again, for folks to check out. Again, I feel like there's synergies. I just definitely want to hit you up, Jake, and have a conversation because I can see some commonalities in the types of problems we're wanting to solve. Our whole premise at Snowpal is one of the things we do, like I mentioned, is APIs is please don't build what you don't have to build. That's literally our pitch. Spend your time and money and energy building it. I'm just pulling up Access Control as an example, but we have other APIs that do things

But the idea, there is some similarities. So we have a bunch of endpoints where you can create these privileges and roles, members and teams and whatnot.

at basically design time and run time, but you configure your system the way you want for it to be. And then you end up using it to run time. So in other words, our approach is different, but we have some things that are in common because our premise is also based on this API at least, is that you can now go create your custom privileges, right? You can create these custom privileges, whatever those privileges are. And then you have a bunch of endpoints to fetch those privileges and whatnot. You can create roles to scope.

those privileges you can have, you know, traditional one, a one is to end relationship. Similarly, you have teams and then you have members and then you can grant access on resources, et cetera. Right? That's the premise. How we approaching it surely is different but I just want to, you know, just to catch you up as well, want to share this. This is just one of our APIs. Now that I've done that, let me.

Jake Moshenko (46:01.726)
Yeah. And I actually think that there's a huge opportunity for providers like yourselves to take all of the code that you wrote to power all of this functionality and just swap it over to use SpicedDB and to use AuthZED. So you have, you probably had a team and they were probably working on these endpoints for a considerable amount of time, right? And it would be great if you could have just made that a thin veneer.

Krish (products.snowpal.com) (46:16.339)
Absolutely, right?

Jake Moshenko (46:27.422)
And now all of a sudden your product can do all of those things and SpicedB is doing all the heavy lifting.

Krish (products.snowpal.com) (46:33.007)
No, totally. The only difference is our APIs not just for us though. Much like yourself, our APIs is for the community as well. So people would use it. So they don't build the permissioning systems. You know what I mean? So permissioning is just one of the things, but yeah, I hear you.

Jake Moshenko (46:44.95)
Right.

But you can do, with AuthZED, you can also do user defined roles, right? You can do what you just showed.

Krish (products.snowpal.com) (46:51.803)
Yeah, no, I'm just saying they are two, I don't wanna say competing offerings, but they are in the same similar space, if you will, right? But that's just one of our APIs. But the idea I wanna impress upon people is.

whatever product you end up using, but I think we are trying to, what we are seeing is the same thing, essentially, at least the way I look at it, is that when companies build solutions, they have to keep the money, the energy, the resources and the time to solving what's unique to the problems that they're actually solving, not reinventing the wheel, right? But that's the examples that we looked at with what you showed, right? Jake was, hey, if you're doing something with permissioning, why do you wanna reimplement it? Don't do it.

just use other systems that actually have solved this problem. Find one that works for you. Whatever it is that works for you, find one. That's the premise. But I want to sort of digress a little bit from the core products, our products themselves, and talk about the more generic problems. So the idea behind, you know, we talk permissioning, but let's say we go outside the space of permissioning. Is that when we, you know, in the next 15 minutes, we want to touch a couple of other peripheral topics, if you will. One is...

Just like your company built something and we built something to solve the problem around permissioning. Is the idea in the future, Jake, is to build systems that are actually not

all inclusive, meaning you're not reinventing the wheel. Let me clarify what I mean by sharing my screen one more time and going to this diagram that we started creating. So in other words, if you took a few years ago, I was feeling people build systems and everything they needed, they actually ended up reinventing the wheel. And I've seen this time and again, even from a managed services standpoint, because people believe that their problems are unique. And I always, we try to tell people that, hey, maybe your problem is unique,

Krish (products.snowpal.com) (48:43.533)
but not every facet of your problem is unique. Maybe you're building something for the pharmaceutical industry, so you're solving a unique problem there. But if you took permissioning, how unique is permissioning for any industry? Even if there are certain aspects to it that are unique, there are many things that are gonna be common. So my question to you is, outside the space of permissioning, like in the future as teams build and solve problems in the world of AI and generative AI and using other managed services,

do you think people should go about solving problems in particular? Be careful about not redoing or reinventing the wheel because there is probably a better solution out there that they are actually better of integrating as opposed to coding to solving those problems.

Jake Moshenko (49:31.806)
Yeah, no, I think that's totally fair. You as a product owner or as a business person should be focusing on innovating in the area that's going to give you a competitive advantage. And you should look for solutions or providers in basically every other aspect of the business, right? You use the term, don't reinvent the wheel. That's totally right. Not invented here is usually the thing that gets settled out, right? Like when people have a...

proclivity to build absolutely everything that's part of their stack. There is something to be said for controlling your own destiny, but when you think about the amount of time that you have in order to get into the market, and that window is always shrinking, right? Like we're getting to the market faster, we're getting to the market cheaper, we're doing things more efficiently, sort of as an industry. And so if you wanna be competitive from that sense, like your first...

Your first thing, you don't write your own database anymore. You just pick one up off the shelf. And that same sort of like, well, we should just be looking for a vendor who does this very well, should constantly be in your thought process every time you make the decision about building something. Because the other thing that's interesting is that if you're building something in an area that's not your core competency, you're going to do the absolute bare minimum, right? You're going to build just enough code so that the thing meets the feature spec.

but you're not gonna get test utilities, you're not going to get fancy DSLs, you're not going to get higher levels of abstraction that maybe you hadn't thought about that a vendor would. And moreover, for every line of code that you write, you have to pay for that forever. So it's like, why not put that maintenance burden, that level of thinking, that ability to seek polish into the hands of a vendor who is really gonna do a much better job.

Does that make sense?

Krish (products.snowpal.com) (51:27.624)
I couldn't agree more. But here's the challenge, I'm gonna pose a question to you. And we are in the same position as well as a company, that's what we tell our clients as well, don't do what you don't need to do because we have a number of APIs published at this point, permissioning that I showed you is just one of it.

But people want to have, like you mentioned, sometimes the urge to have that level of control. I've seen that even with a very fantastic engineering team. And actually, the problem is more aggravated, if you will, the better the engineering team. Because now they're like, you know what? I can actually do this in six weeks or in a couple of months or whatever the time frame is. Why should I actually go leverage another provider? There's a two-part question. What would be your answer if I were a developer?

4 weeks I'll come up with something that will work and it might just fit our needs for today even if not till the end of time or till eternity. The second part of the question is, one is for me to get that working and two is how much can I trust the other company. One if you're going with a really large organization that's been around a long time, people have a bit more trust because they feel like they're going to be around longer.

with the companies being small. Again, there's pros and cons with larger and smaller companies, but just playing the devil's advocate. People have asked us, hey, you know what? We've actually changed our licensing models, Jake, to some extent to satisfy the needs of

different sizes of clients. That's why my second question is not manufactured. It's a very genuine question. The question is, okay, you are a startup. Can I trust my data with you?

What what is my plan B if you guys have some change you have your interest change your funding situation Changes or whatnot. How do I go about it? Right So my two questions are one is if I'm a good engineer if I'm working in a team of great engineers What do you tell them saying if someone told you I can do this? I know your product does a lot more but I only need like 8% of it and I can build 8% I want what is your answer to that question and the second question just to shorten what I was rambled earlier was

Krish (products.snowpal.com) (53:36.843)
Trust factor in not just the data, in the functionality, the uptime, the services and all of those things. How do you address those two questions? Not necessarily from an odds-z standpoint, just I wanna see what your answer would be for any more generically, yeah.

Jake Moshenko (53:49.822)
Yeah, generically.

Yeah. So, why would I choose to buy something when I could build it? I agree with you, the more talented the engineering team, the more confidence that they have that they'll actually be able to ship something. And that is a challenge for sure. I think it goes back to what I was saying earlier, which is like, if you do build something, one, you have to pay for it forever. I was reading a stat the other day that per thousand lines of code,

that get written, there's something like 45 bugs and 15% or 15 of them make it to the customer, right? So like, why do you want to be spending your time on that? That's not a core competitive advantage. Someone else has already hit those bugs. Someone else has already solved those bugs. So like, the other thing I like to throw around or we like to throw around in our org is that any given piece of software, there's the first 80% that you work on, right? Which is all the fun, meaty, hard parts of the problem. And then there's the second 80%.

that you work on, which are all the things that you forgot about, the documentation, the continuous integration, the testing, fixing the bugs, the ongoing maintenance. And what do you do when the person who wrote that code leaves the company? And it's not really something that you want to be doing. So these are all great reasons to bring in a vendor. You're going to get a more complete solution, a better solution, a longer lasting solution. That's purpose-built for the actual task at hand. And then the second part of your question was, why would I trust a startup?

Krish (products.snowpal.com) (55:12.747)
Perfect.

Jake Moshenko (55:17.146)
I can't answer that super generically because I think it matters on a case by case basis. But what we recommend that people do, well, one is that we're open source. So SpicedDB is open source. So we always can say, hey, if we disappeared tomorrow, which by the way, we're not going to do, but if we disappeared tomorrow, it's open source. You can still run it. You can still get all that goodness. The code doesn't change. It's very liberally licensed. Just pick it up and run with it. But even beyond that, there are reasons to trust a startup.

Startups are doing things quicker, more responsively, more innovative than the big companies. And so if you want access to that, if having access to that innovation gives you a big enough edge, such that you can still reap a risk adjusted reward, then it makes sense to go with a startup. Does that make sense? The risk adjusted reward part of it?

Krish (products.snowpal.com) (56:06.935)
Yeah, it makes complete sense. Uh, and it, I think both answers make a lot of sense, but I want to reiterate on the first one, which is, I think if you end up doing what you don't have to do to prove a point to yourself or to your clients out of the market, you lose that time. It's not just a question of money. You lose that time. That's the most precious asset any of us has other than probably water or maybe with alongside with water. Those are very precious. So I think it's important to ask the question. And I, I say this because.

because having done this a long time, much like Arsilejic, I've seen people want to do it. It's not so much, hey, I don't trust the other company or the other product. It's, I think a developer mindset, if you will.

it's saying, hey, if I can do it, why don't I do it? But I think I've personally and professionally tried to shy away from that because you want to give people the benefit of the doubt. You do your due diligence and see what they do best and integrate and incorporate them into the mix of things. And as we speak about this and have this conversation in 2024, there's a code that a lot of us don't even have to write with the tools.

chat, GPT and whatnot. I think nowadays, you go to the editor with Copilot. I know the other day I mentioned this in a different podcast, it's not even code. I was writing some Go APIs and I wanted to add a comment and I started.

I didn't even type the comment. And we're not a team that says get restaurants is returning restaurants. We're not stating the obvious at all. We only comment about the business, not about the code. And I sat there for a second thinking, is this thing really smart that it can actually generate the comment that's in my mind? And I'm not making this up. It was actually a very specific business comment that I wanted to write. And I waited two seconds because I was lazy to even type it to say I want to let Copilot do the job and paying 20 bucks after all.

Jake Moshenko (57:41.506)
Yeah.

Krish (products.snowpal.com) (58:02.679)
it actually generated the comment exactly the way I would have written it. And I looked at it for the next 10 seconds, admiring the comment because I was like, this thing is beautiful, which tells me that software needs to be built very differently going forward. You know, however you did it in the last many years, if, you know, I think I don't know what the right or the wrong ways, but I think personally and professionally, the one thing that I feel is going to be the wrong way is

Jake Moshenko (58:10.254)
Hahaha

Yeah.

Krish (products.snowpal.com) (58:29.291)
Krish, if you're solving a problem today, like you did it three months ago, you're actually doing it wrong. That's it, that's my premise. I wanna start there and convince myself that it's still there is value add in solving the problem the same way. So just adding to your points, very valid points. When you look at a problem, see who's solving that better.

Jake Moshenko (58:36.406)
Hmm. Yeah.

Krish (products.snowpal.com) (58:52.367)
look for companies, engage those companies, do your due diligence that's on us as providers and then people who are creating it, which makes great sense. So I think thanks for showing, you know, oddZ, at least, you know, you spent I think 15, 20 minutes, maybe a little bit more on the product, you know, at least the actual interface. So people have a starting point to look at it, to check out, and I'm gonna include the links in the podcast oddZ and alongside your

LinkedIn profile and any other links that you want me to include, we are happy to include that. Jake. So in terms of closing comments, anything you want to sort of say to, to wrap up this, this conversation around permissioning the company, the topic, the broader items we talked about. Cause I like to leave and the podcast with folks hearing the guests wise and not my voice. So anything that you want to close out with.

Jake Moshenko (59:46.218)
Yeah, I'll just say that if you found the type of conversation that we had and the kind of modeling that we did for an application interesting, you can have that kind of chat with me. So we have a special landing page just for podcast listeners. It's at AuthZed.com slash podcasts. And you can go and schedule a meeting and we can talk about what permissions look like for your application. Right. So we can, we can have that chat about whether it's something that you should be building or buying.

So, I mean, that's all. And then I guess since you brought up Copilot and how long you had to sit there admiring the comment, I will say does chat GPT actually save us time when it generates something very fast, but then it gives us a couple hours of existential dread after we see the output, right? So I don't know.

Krish (products.snowpal.com) (01:00:35.755)
There's no question about that. So the beauty is, I want to end this with your wise, but I'm going to say this, I think to myself.

You know, there's two ways to react to tools and progress. And I've seen great people react quite differently. One is you have the selfish initial reaction, which is, this is going to change my life, my lifestyle. Am I going to have to do things differently? I'm going to have a job. They're all genuine concerns. The other side is the positive, the way you look at it, at least in my mind, you have more tools, you can do more. We all have a finite time on planet earth, right? That's finite. Whatever the time is, it's finite.

So the more tools you have, the more you can actually get done. To me, that's the positive. You didn't have cars and planes. You probably could travel to one country taking a boat or a ship from 100 years ago. The fact that you have these planes, even though they're not as fast as they should possibly be in this time and age, you can go to more places. So to me, that's no different from an engineering standpoint. I think you have to, I think there isn't, there's one way to look, there's multiple ways to look at existential threads. But.

The way I see it is you have more tools, the more somebody else does for you, the more time you have to do more of what you can possibly do and bring more value add to the table. So this is an exciting time, I think, to be alive as a software engineer. It's just there's multiple reactions on a daily basis. This was a great call, Jake. I would love to have more conversations around not just our products personally.

But even any topics that are of interest as well. But I appreciate you taking the time, Jake. Before I end this one, again, folks, Jake Moshenko is an innovator in the cloud native ecosystem. Definitely check out Jake's product, AuthZed, the company commercializing

SpiceDB. And if you have any questions, I'm going to include all the links that Jake's going to give me. And definitely hit up Jake and team and go from there.

Krish (products.snowpal.com) (01:02:35.187)
This is great, Jake. Again, thank you so much for your time.

Jake Moshenko (01:02:38.498)
Thanks for having me.