Interviewer:
So you're the head of data, right? Or at City Storage Systems or actually no, you were the head of data. Now you're the head of infrastructure. Right? So you recently were promoted.

Interviewee:
Yeah, yeah, as of last week.

Interviewer:
All right, well, congratulations. Well done.

Interviewee:
Yeah, thank you guys.

Interviewer:
So what's the new role? What do you got to do?

Interviewee:
Pretty much the same as... Well, when I think about infrastructure, I think how many of you had water today, drank any liquid? Yeah. I'll be surprised if somebody didn't. We can survive three days without liquid, without water. How many of you actually thought about how did water get into your cup? Nobody. It's kind of interesting. We can survive for three days without it and then we never think about how complex this whole infrastructure is. And it's like, in United States it's easy. In some countries it's really, really hard. Now going back to software infrastructure, it's going to be controversial interview by the way. So we tend to say it's all awesome, amazing cloud providers, like one click wherever. We all know it's not. It's not. It's complicated. It's hard. Stuff fails. Systems are complex. And when I think about infrastructure at City Storage Systems, that's actually vision, right, for our teams, which for our business verticals, we should be invisible. We should be as needed as water and we should be as invisible as all this real world infrastructure that delivers water. So I guess that's a challenge.

Interviewer:
Yeah, I mean, yeah, even think about the Romans. It took them building all these huge systems of pillars and all kinds of things to get water to people. So yeah, a lot of complexity with infrastructure. All right, so you work for City Storage Systems, exciting name, what is it? What do you guys do?

Interviewee:
[inaudible 00:02:06]-

Interviewer:
You store all the Raid for cockroaches that are in cities, or what do you guys do?

Interviewee:
So our founder, you may or may not heard of him, Travis Kalanick, who built Uber. He thinks of Uber as a network. If you think about city as a computer, as an Uber, the transportation is a network which connects something. Network usually carries data, right? What does network connect? It connects storage. That's why company name is City Storage Systems because what we try to do, we try to take spaces in the city and add technology to it. That's as best as I can explain.

And then to the good example would be you mentioned CloudKitchens. It's something we known for. CloudKitchens try to solve a very complicated problem. Companies like Uber, it's made food delivery possible like last mile. If you think in software terms, it's like CDN and that's HCompute because ingredients travel into a restaurant, they convert it into food, ship to you, but that last mile is very expensive. I don't know if you realize how expensive is it, but it's very, very expensive and we don't believe it's sustainable. So what we're trying to do, we're trying to build infrastructure for restaurants so we can drive delivery price and actually not delivery price. The price of food to a level when anyone can eat using this restaurant quality food every day, three times a day, that would be awesome actually.

Interviewer:
Awesome. That's really cool. And then I think the thing we're going to talk about today is toil, right?

Interviewee:
Yeah.

Interviewer:
And I'd love to get your perspective on this. So how would you define toil or what that means for City Storage Systems?

Interviewee:
Yeah, I try to make the title as boring, as uninspiring as it possibly be, but the topic of toil is very interesting. So I personally define toil is anything I don't want to do. Right? And if you think about it, let's talk about engineers. We hire really talented people to do exciting work. We want innovation, right? Every company, probably if you heard when execs are talking, is like, "More innovation, but faster, faster innovations." I guess that's the thing. But when you're doing things, going back to water, when you walk into a river with a bucket to get the water and walking down back down, you spending all this time, you need a lot of water, you need to walk many, many times. You want the pool that becomes actually incredibly long. And toil, in my opinion, is not just outages or writing documentation, it's even thinking about core optimization.

If I want to build a software because product wants me to, every time I think about infrastructure, we fail as infrastructure. Right? Every time I'm like, "How do I deploy? Which is... Is it TC2? Is it Kubernetes? Who cares? Where is my database? Which cluster?" Right? It's introduces delays. And if you think about, again, going back to computers, the way computer works, there's CPU, work comes. If work can be done in CPU and memory, it's fast. Every time you hit IO, it's a losing game, which is another controversial topic because we're talking about distributed database where network exists.

But with people with us, it's the same problem. You hit IO, which means you need to talk to somebody, you need to think about something that is not in your cache. You need to read documentation, you get on slower, you're going to make mistakes as a human and results will be probably suboptimal. And you can't go and talk to my storage team because we are way smaller and again, we hire only engineers, we don't have operations. And engineers don't necessarily want to be consulting as different companies do differently. Engineers want to write code, right? Not to go and say, "Well, yes, [inaudible 00:06:32] query looks funky," Right? You should just reorder that job for a machine. That's why I think toil in general, it's an interesting topic because it's a topic of making systems work as they intend them to work, not as they work today.

Interviewer:
Interesting. All right, so yeah, so you have... The way you kind of counter this is just through engineering, right?

Interviewee:
Yes.

Interviewer:
Not having the operations team to actually-

Interviewee:
Operations.

Interviewer:
... do all that toil thing?

Interviewee:
Yeah.

Interviewer:
So the engineers are kind of responsible for building the automation or building the things that they don't want to do or don't want to operate or were support to make sure that things are always on and water's flowing?

Interviewee:

Yes, that's correct. Yeah. Technically, we never ask engineers to build automations because we ask engineers to make an area workable. Meaning, if engineers, like in your case, decide that to fix a problem, we need to build a new database. And we did it actually, which we made... We started blog post and we're probably going to talk about it. It's not a production. It's not a old TPL app, but it's something interesting. And if engineers think that we can solve a problem by is fixing existing database, right, not everything can be solved around the system. Sometimes you need to go in the system, or you start completely from scratch. It's also what engineers do and it's the right way to do it, I'll believe.

Interviewer:

Cool. Well, that's awesome. So let's talk about this from your journey with us for the past four years, right? Because there's things, just knowing a little bit about you guys. I know early on you guys adopted Kubernetes in a time where we didn't have a lot, we didn't have an operator, we were just getting started with it. A couple of customers were adopting it. We were actually teaching customers how to use Kubernetes for a while actually with Cockroach. So you adopted it really early and you had to build that whole operational experience with that. What was that like in the early days?

Interviewee:

So Kubernetes, it's one of very little technologies which we actually get from the cloud provider. So you can by extension kind of assume everything else we do ourselves. Not everything else, but many things. And Kubernetes is a very complicated system actually. I almost feel right now, the advertising page now, containers are easy, it's super easy, it's a little bit deceiving. It's actually very nuanced. It's hard, but it actually ended up being a really good bet for us. The reason is our early vision was that we want to be multi, multi anything. Multi-cloud, multiregional, multi-AZ. When Musk finally gets to Mars, I guess we'll be multi-planetary, multi everything. And then if you think about it, I mentioned City Storage Systems. I talked about physical locations and physical location, it's kind of a room. It may or may not have rack of servers, and I'm not saying we have one, but Kubernetes can also run on all those different types of systems.

And having a single way to manage applications was very appealing to us because we thought we're going to bet on Kubernetes because we believe that industry converging on Kubernetes, and if we do it, we'll simplify our lives if we were to expand to another cloud, expand into a physical location, right? Because it's not going to be as many disruptions if we pretty much look to one single cloud provider. And then there's going to be one of those stories of, it took us two years to move from A to B, and I don't know how many millions of dollars.

Interviewer:

Okay. Yeah, let's talk about that. And I think you've done this, right? You've moved between cloud providers as well?

Interviewee:

Yeah, I did it actually two times in the last five years. First time, it was very simple stack. I think it was MySQL database, we had to do some, it was really early and it was very simple. Second time, it was a major migration. What was really interesting is that we were given a deadline, we were told pretty much, "You guys have nine months and you have to be in another place." And we're like, "Oh, cool, can we even run stuff there? I mean, did anyone do a load test? What about... Are disks the same? Your CPU performance similar? What about networking?" It was interesting journey. I learned a lot about multi-cloud. Well, I had a theoretical experience before. That was my practical experience, not my first time expanding into regions.

Interviewer:

And how close were you between the theoretical and the practical?

Interviewee:

Very close. The problem with multi-cloud, well, again, in our case we had Kubernetes, so we kind of solved most of the problems and we also had another, not a mandate agreement amongst engineers that we are not going to be cloud-locked because we're not cloud-locked. As you imagine, all of our software, all of our info was compatible with wherever is on the other side. There were only couple systems which were really needed from the cloud. And some of them gave us surprises actually, which is topic for another talk on a different conference and a blog post and it's all going to be coming. But actually expanding into the cloud and moving, let's say Cockroach, was very, very simple. The things that get you is costs, is network, right? And network is expensive between clouds. Sometimes prohibit very, very expensive.

Interviewer:

Okay. Yeah, no, that's fair. But it sounds like the portability experience was good though.

Interviewee:

Portability. Yeah, especially with things we built. It was very boring migration. We expanded and we contracted and that's pretty much it.

Interviewer:

Cool.

Interviewee:

Rest of the data stack was fun. Like Blob, moving Blob storage. Oof. And all the systems on top of Blob, also Labstack, streaming. Yeah, that's a next-level complexity.

Interviewer:

Again, another blog post, another conversation.

Interviewee:

Yes, yeah, that's coming.

Interviewer:

All right, cool. Well, let's go back to the Kubernetes deployment, right? So this conversation is about toil too, right? And how did your team, how'd you guys organize yourselves to do all those management pieces for it, or how to do provisioning, how to do bootstrapping and onboard or even getting applications and teams onboard with how Cockroach was going to work within Kubernetes?

Interviewee:

Yeah. Very early we started with Helm and then I think, I don't remember whether we used Helm Operator for Cockroach, but that was absolute beginning. Then after many hours of arguing internally and disagreeing at the end, so different teams did stuff slightly differently. Storage team decided to double down on Kubernetes operators and went very, very deep into them. We built our own operator. It's multi-cluster.

What it means that it technically controls many Cockroach clusters stretched across geographies. And when I say operator, if you guys know what Kubernetes is, though it works with Kubernetes, you give it a YAML and it does stuff for you. The question is what's in a YAML? And you can go simple where you say, "Build me a database." You can go really deep when you can say, "I want a table." Right? We went that deep. So our YAML is complicated, but it also means that amount of manual work, the teams that engineers need to do was minimized. If we have a problem, we write code, we test code, we deploy, and we never talk about this problem again.

Interviewer:

Cool. What about-

Interviewee:

CDC as well is same. It's operator. We say, "Okay, here's Kubernetes resource. I want table A, be replicated to Iceberg," in our case or Houdini, we use both. Go.

Interviewer:

Okay, awesome. What about the type of workloads that you support in the environment? So there's tons of workloads out there that could be transactional, search, OLAP, what have you found as to be a good fit for the workloads that are running in your environment?

Interviewee:

OLTP, right, of course. So transactional workloads. Now it's a deep topic. When we started, we didn't have an army of engineers, so we had to prioritize. And I do believe that we started early enough... Yeah, when we started using Cockroach, CDC wasn't there yet. We had another blog post about a queuing system built on top of Cockroach actually. And original proposal was to use CDC streams. And after some testing, we figured out that they were not up to your standard. And we just, we worked with you guys of getting them there. But during this time, we thought that probably using CDC to replicate all the data is not the best thing to do. So what we did, again, controversy, we use Trino, is our main distributed core engine for OLAP. And one of my early colleagues came from Trino team, which was really helpful, right, because we could actually, when we needed, we just implemented right inside Trino the safeguards and wherever we need it. But in many cases we just said, "Okay, this Trino, you can just query this production Cockroach table." Super controversial.

Now we only had one trouble in production because of it, and there was a trouble because, and a little bit technical, the way Trino executed the query, it wasn't using as of a system time. So it got into a place where it was actually conflicting with live transactions. And then the problem was that Trino didn't have a connector for Cockroach. But as I said, because we had a commit, not committer, person from a Trino team, we just implemented this really fast and it ran for probably another year and a half, no problem whatsoever in production. Analysts just hitting technically tier zero databases till we got a little bit more manpower and CDC got to a place where we felt comfortable using them and just will move everything to CDC.

Interviewer:

Cool.

Interviewee:

Right, but that's on a lab, so it's possible if you know what you're doing. Actually, I don't think we open-sourced the connector for... Maybe somebody else did, but that's one of the things. And then of course you can do rate limits and stuff. Time series we never tried. We have specialized databases for time series data search, same. ETL, also no, no, not.

Interviewer:

Okay.

Interviewee:

But of course, there's a problem going to water. When you give an engineer an SQL interface, it opens Pandora box. So for 10 years folks, I thought SQL is that, right? I worked on Apache Cassandra, I built almost my own database for Sony PlayStation, which built like a database per user. Right? And I thought, "Okay, I want this... As an engineer, I want this very tight control of what's going on. I want exactly know if there's issue, I get. I want to know how much traffic is going to generate, what's happening on networking stack, on everything."

Because then I could optimize. And now as a provider of the infrastructure which has SQL on top, it's actually scary because engineers can do anything. Like today, we were discussing with the team and somebody like, "Oh, we have that many instances of recursive queries," or, "We have that many instances of using window functions." And we were like, actually, I don't know if anyone overheard and we're like, "Should we just limit SQL? Should we just say, 'All right guys, you're free to do whatever, but we're just not going to let you do something'?" So yeah, that's how it goes.

Interviewer:

Yeah, that's why I asked. If you have an organization full of engineers and you give them a SQL database, that you think you could do everything with it. So I know a lot of our customers push us, right? They want us to be more general purpose or be able to do all kinds of workloads and so forth. So I was wondering if you were kind of getting that sort of energy or that sort of interest from the folks that you support. That's why I asked the question.

Interviewee:

We do. We do. And then the trade-off is there when it's super flexible. It's super dangerous. Again, infrastructure is not like water. You have to go with a bucket and sometimes if you do something really bad, your bucket will not have... Well, it will be empty when you come back and that's how it is.

Interviewer:

Cool. What about for mission-critical workloads? So do you have Cockroach doing things differently for some of your mission-critical workloads versus your non-mission critical workloads?

Interviewee:

Yes. Yeah, more interesting stuff. We are multiregional, but we only run one availability zone per region. Why? Because we did mass and apparently AZs fail all the time, but they very rarely fail into different regions. So if you're already stretched, and if you think about it, we're on Kubernetes, so we use regional attached disks, right? So those disks are replicated I think two, three times depending on configuration. Then we have I think five replicas for multi-regional clusters. And it's already stretched so many times what additional zone give you, we decided that it doesn't give us much. So that's our topology. For a single, for a non-mission critical, we'll literally run them in a single region, single AZ. And statistically, we didn't see many outages when the whole cluster was wiped out. Sometimes it happens, but again, it's just mass, how much money infrastructure, it's only, it's just dollars.

How much money do you want to spend, right, for what you're going to get? If it's non-mission critical, which means it's a system that nobody, not nobody cares about, but if it dies in the middle of a night, nobody going to cry on page. Yeah, okay, it can die sometimes. We do that. We tried running on spot and my advice, if you try running on spot on Kubernetes, make sure cloud provider can register those spot nodes cleanly because if cloud provider can't, what's going to happen, the VM going to die, but Kubernetes will still think that node is alive. And it's really hard to reschedule this workload because you cannot unmount disks in some limbo and that wasn't a very positive experience. So we stopped. But I think on VMs it's probably more, less controversial. Or with a Kubernetes that can clearly get a signal, sport is going away and actually say, "Okay, yeah, it's gone." We were even thinking to build some on automation to detect the situation and actually force remove, but it's not as simple.

Interviewer:

Nice, that's cool that you guys try this stuff. Pretty neat. What could we do better? What could Cockroach Labs do better for City Storage Systems?

Interviewee:

Yeah, I think-

Interviewer:

Or even our community or our customers. What do you think we could do better?

Interviewee:

Yeah, I think we have this very good thing going. Again, 2.4 introduces a lot of very interesting fixes, especially because as I said, there's I think five fallacies of distributed systems. Network is... Latency is zero, bandwidth is infinite, network is reliable. And once you cross network, once you go into network land, weird stuff can happen. We saw a lot of weird stuff. Asymmetric partition really appreciates the fix, which promoted the symmetric into symmetric, partial partitions, and all those new features which we may or may not contribute it to. Hopefully, we had some positive impact as something we're really looking for. We try to squeeze every single drop from our infrastructure and our databases.

For example, we run on Kubernetes, we auto-scale CPUs for pretty much everything, including Cockroach clusters, meaning we give Cockroach as much CPU as we think it needs and no more. We auto-scale disks. Our auto-scale is probably, we not there yet of the scaling, but we have a way to, in a very easy way to resize disks both up and down so we don't overspend. And doing all of this crazy stuff means that our Cockroach is usually runs in a very uncomfortable setting. It runs either hot or IO is restricted, right? And all the contributions around core functionality and reliability are greatly appreciated because as you probably notice if we were talking about restricting SQL, we're not necessarily into, let's do more features on top of the database, but making the core engine very, very, very, just impossible to kill.

There's one thing and another thing, which is very interesting domain to me as well is, which seems simple, but it's very complicated. It's helping engineers not to think about database. Automatically... So core optimization is a very complex topic. Core optimizers, as you know, are not actually precise. There's usually tight time budget. Optimizer can do stuff human wouldn't do. But it all starts with developer writes some code. They don't understand database and it's not their job to... I would say learning database really deep for them. It's a waste of money for the company because there's my team that's supposed to do it and we're supposed to provide something to them.

But there is things which we just don't have visibility into. Questions as, I have the SQL query, how expensive is it? What's the CPU budget, memory, IO budget, network, how... Or there is a query coming. What's a trace of the query? What notes will hit? What ranges got hit? What happens if this specific set of ranges go down? Can application protect themselves if system is partially unavailable? Because when system is strongly consistent, partial unavailability can mean full unavailability on a system of top. So put in more wood behind that arrow would be greatly, greatly appreciated.

Interviewer:

Yeah, so the whole query developer experience, right?

Interviewee:

Query developer experience. Yes. And then making multi-tenancies interesting concept. Almost like making... Almost... Engineers should not know, ideally my group exists or Cockroach exists, right? This stuff works. [inaudible 00:28:04]. And they say, "Safe, it's done."

Interviewer:

Awesome. Well, listen, I appreciate that you guys have given us a lot of feedback over the years. I remember we were pretty instrumental where we were going with multi-region. We appreciate all the support there and the insight of where you wanted things to go. So yeah, we can definitely look at the developer experience stuff. We'll continue to focus on reliability. That's core to what we do.

Interviewee:

Yeah, it's actually fun thing. We discussed the term super region, right? We discussed it at 2020 early because we ideally want it to be global and then we figured out it's going to be too expensive. So we thought, "Okay, we're going to stamp region per continent." Called it super region.

Interviewer:

Yeah.

Interviewee:

Yeah.

Interviewer:

Cool. Well, listen, I think we're just about out of time. I want to appreciate, like I said, I appreciate you coming, taking the time to talk with all of us today. Wish you much success in the future and thanks again for being here.