EPISODE 844

[INTRODUCTION]

[0:00:00.3] JM: Each cloud provider offers a different set of services which are not always compatible with each other. What are the challenges of building an application that interoperates with multiple different clouds? The first issue is API compatibility. To get these different cloud providers to interoperate with each other, we need to have compatible APIs. Most cloud providers do have a managed SQL offering and a bucket storage system and server abstractions, like virtual machines and containers, but these tools might have different APIs on each cloud.

The code that you wrote to save your application data to Amazon S3 might have to be rewritten if you decide to switch your bucket storage to a different provider. Another issue that interferes with cloud interoperability is the degree of integration on a particular cloud. If I build my application for AWS, I might be heavily integrated with Amazon's identity and access management policy system, or AWS logging. Each cloud provider makes it particularly easy to connect with their vertically integrated solutions. There's also the problem of services on one cloud that simply do not map to a service on any other cloud.

Google Cloud BigTable does not have a directly equivalent service on Amazon. Microsoft CosmosDB does not have an equivalent service on DigitalOcean. As developers, this interoperability irritates us. We want to be able to deploy our application to any cloud. We want to be able to move applications easily from one cloud to another. We want a straightforward failover strategy from one cloud to another and we want to mix tools from different clouds together as easily as we import libraries in languages.

Bassam Tabbara is the CEO of Upbound; a company that's focused on making multi-cloud applications easier to deploy and operate. I spoke to Bassam at KubeCon EU 2019 and he described the problems of multi-cloud deployments and the opportunities for the cloud native ecosystem to become more cross-compatible. Bassam is a great guest. He's been on the show before. I think you'll enjoy this episode.

Some upcoming announcements; we've got conferences I'm attending, Datadog Dash, July 16th and 17th in New York. I will be there. The Open Core Summit, September 19th and 20th in San Francisco. We are hiring two interns for software engineering and business development. If you're interested in either of these positions, you can send me an e-mail with your resume to jeff@softwareengineeringdaily.com. Put internship in the subject line, preferably.

FindCollabs is the company I'm building. We launched several new features recently. If you have a cool project that you're working on, I would love to see it. I check out every project that gets posted to FindCollabs and I've been interviewing people from some of these projects on the FindCollabs Podcast. I am happy to see your projects on FindCollabs if you post them.

We have a new Software Daily app for iOS. You can become a paid subscriber for ad-free episodes at softwareengineeringdaily.com/subscribe. If you've checked out the iOS app before, I recommend giving another shop. We've made some great updates recently and Android updates are coming soon.

With that, let's get on to today's show.

[SPONSOR MESSAGE]

[0:03:52.8] JM: Deploying to the cloud should be simple. You shouldn't feel locked in and your cloud provider should offer you customer support 24 hours a day, seven days a week, because you might be up in the middle of the night trying to figure out why your application is having errors and your cloud provider's support team should be there to help you.

Linode is a simple, efficient cloud provider with excellent customer support. Today you can get $20 in free credit by going to linode.com/sedaily and signing up with code SEDAILY2019. Linode has been offering hosting for 16 years and the roots of the company are in its name. Linode gives you Linux nodes at an affordable price, with security, high availability and customer service.

You can get $20 in free credit by going to linode.com/sedaily, signing up with code SEDAILY2019 and get your application deployed to Linode. Linode makes it easy to deploy and

scale those applications with high-uptime. You've got features like backups and node balancers to give you additional tooling when you need it. Of course, you can get free credits of $20 by going to linode.com/sedaily and entering code SEDAILY2019.
Thanks for supporting Software Engineering Daily and thanks to Linode.

[INTERVIEW]

[0:05:29.6] JM: Bassam Tabbara, welcome to Software Engineering Daily once again.

[0:05:32.3] BT: Thanks for having me again.

[0:05:33.2] JM: Thanks for coming back on. We've done several shows. I like talking to you, because you have a deep perspective on the different cloud providers, but you are not from any particular cloud provider. Your company Upbound is about managing cloud compute resources in a cloud agnostic way. Can you give me the vision for how you see people managing compute resources in two, five, 10 years, however long you think it'll take to get to a place that satisfies the minimum version of your vision?

[0:06:19.0] BT: Sure, I can try. I mean, essentially, the way that we like to think about cloud computing going forward is that we as a community can arrive at a more open cloud computing platform, essentially gives people more choice. What does that mean? Let's unpack that a little bit.

I think, if you think about every cloud provider today, the way that you consume their services is that you go talk to their proprietary control plans, you're using their API, you are deploying and manage services on their platform. Whether you're using open source or not, you are always going through their proprietary control planes and using their infrastructure.

We think that the world going forward should look more like, a more open cloud platform that lets you deploy and run infrastructure wherever you want it to run, without having to essentially retool, re-engineer your applications and services every time you want to make, run on a different environment, or on a different platform.

We think portability is interesting. We believe that there needs to be a more open cloud computing platform. Both of those things are interesting from the thing that excited about what we're doing at Upbound and what we're doing in general in the open source community.

[0:07:40.8] JM: That sounds not dissimilar from Google's cloud vision. They say we are the open cloud. How does your vision differ from theirs?

[0:07:50.6] BT: Yeah. I mean, I think there's a lot of different companies that are talking about open cloud and now multi-cloud is a thing. I think the thing that's interesting to us is that we think that if you are a cloud provider and you're spending 10 billion dollars in CAPEX every year, and the business model is such that you are making money on renting compute hours, you're at odds with essentially being a cloud agnostic platform for computing. We think that it needs to be more of a community-driven effort and one that's not tied to running data center business.

At Upbound and some of the projects we do, we're not in the data center business. We're not in the CAPEX-intensive running data center business. As a result, we take the side of the enterprise and the community in terms of what enabling more choice, enabling the best deployment, doing the best policies for where things should run. We think that that's actually a better place to be then potentially leading with something that is multi-cloud, but really the end game is to run on one cloud.

[0:09:09.0] JM: When you start a company, you do have to have a vision, which the vision you had when we first talked a couple years ago is consistent with what you have today. I think you got the vision right initially. When you start a company, you also have some initial set of tactics that you're going to execute on. Those tend to change. Those tend to be more adjustable than the vision for the initial company. How have your tactics evolved as you've started to implement this strategic vision for Crossplane and Upbound and your multi-cloud vision?

[0:09:52.0] BT: I mean, I think one of the things that we knew early on is that we should lead with and we should be active participants in a community-driven effort, around the vision of an open cloud, in a world where you have more choice around cloud computing. We didn't really know what that meant when we started. We had this Rook project, which is really successful. We've been working with the community. We learned a lot about community engagement and

how to drive a healthy community around the Rook project. We thought that we should go create a set of different pieces that enable us to arrive at a more open cloud computing platform.

There are things like Kubernetes and other projects around that are super amazing and super useful. We thought there were some missing pieces, but we didn't really know how to essentially position them. Is it one project that's going to do that? Is it multiple projects? We organically found our way and to starting the Crossplane project, which we thought was a centerpiece of any cloud offering as a control plane. We pushed that out on a mostly, here is the vision, here is essentially a POC at version 0.1 that brings it all together, but is true to the vision. We at this point, are now iterating with the community on this. I think that's an important step in the sense that you're not all in a room, you're actually working all together in the open, trying to get a project to get a certain level of adoption.

In terms of what tactics changed, honestly I think we're trying to put things out early and work with the community, because let the community drive where we want to go. As opposed to working out in isolation and trying to figure out okay, here's the grand strategy and here's the grand plan. We're learning a lot from being early and in this space and from engaging with folks on these projects.

[0:11:56.2] JM: It's a strategic template, I think, where you started with the broad vision, which is something that people absolutely want. Plus a very specific project, which is Rook, this storage interface for using – I mean, I believe the initial spec is being able to use SEF with Kubernetes, right? SEF, which is this way to manage storage in a cloud agnostic way that is difficult to use, if I understand correctly. Rook was a shim over it, that made it a little bit easier to use and more specifically, easy to integrate with Kubernetes. Is that accurate description of Rook?

[0:12:40.5] BT: Yeah. That's roughly how it started. We realized that there was plenty of data plains for actual storage, putting bits on disk. What was missing is essentially an orchestrator for them that helps them integrate into cloud native environments. We started with Rook and built an orchestrator for Rook. Now we've got seven other storage backends as part of Rook that

we're orchestrating and more coming. Essentially, Rook became data control plane for the cluster.

[0:13:11.9] JM: Right. I want to set a strategic template. Well, I think you have with the Rook project, plus the vision for the company is okay, we have a vision for what we want to accomplish. We've got an open source project that is a small sub-component of that vision and we're going to just play in the space, talk to people, see what happens. It's how the Linkerd team started, the Buoyant team.

They said, “Okay, we've got Finagle. We think this probably belongs – something like this belongs somewhere in the ideal stack. We wanted to help people with microservices, or whatever you want to call these cloud native deployment schemas. We don't really know what the business model is going to be. We don't really know what the product is going to be. We've got an open source project. We're going to nurture it and hopefully, it turns into something.” That seems like not a bad strategy for this market, where you've got venture capital dollars sloshing around, looking for cool projects to invest in. You've got technical teams that are passionate about their community working on an open-source project, with no vision for making money anytime soon. That's actually a really good strategy.

[0:14:30.2] BT: It's a great strategy. I also think that you should definitely be thinking about not only the vision, but also how do you actually make money as well. You should think about – you should have an idea. You should have an informed idea of where there is a business and how to get there. You might be completely wrong about it and you should be totally open to experiment and iterate. Not having the idea means that you spend also a lot of time building things that are engineering for an engineering sake, or people that the customers don't actually want. Or maybe you find a niche of users that tell you they really want this thing, but nobody wants to pay for it, or nobody – it doesn't lead into a business.

I think if you define a success of a project as just community adoption, then that's great. You should also think – I think about how do you arrive at a business around something, because that's when it gets to the next level. Your point is interesting, but I don't think you should ignore the fact that all this VC money is in it also for essentially building healthy businesses around a vision and teams.

[0:15:48.3] JM: Right. The thing you and I were talking about yesterday was that first of all, you built Crossplane. We did a show about that in the past. Crossplane is a – honestly in that show, but basically in our last two shows, both times have been somewhat confused by what the heck you're doing, because it's a little – it's technical and it's been hard for me to wrap my head around. Crossplane is a multi-cloud control plane.

I wasn't really able to understand it from that show, although people can go back and listen to it. Maybe they'll understand better than I did. When I heard the example that you told me last night, it actually made a whole lot of sense. We were talking – this is public, right? The GitLab thing. Okay. GitLab can deploy GitLab itself using Crossplane. I want to describe this example in more detail. First, in order to describe that example in more detail, I think a lot of people have heard about GitLab, but they may not understand exactly what it is and how it gets deployed. Explain the typical deployment model for GitLab.

[0:16:55.4] BT: GitLab has essentially a helm chart. They run on Kubernetes. They have a helm chart that essentially packages all their configuration for all the deployments and replica sets and all the Kubernetes –

[0:17:12.0] JM: GitLab is self-hosted git.

[0:17:13.7] BT: Pretty much. Yes. A typical customer of GitLab will say, “I pick a cloud provider of choice.” They'll say, “We want to deploy GitLab there.” It looks like, something let's say it was AWS, right? The customer would light up an EKS cluster, run the helm chart for GitLab, deploy all those things, they'd have to go provision a database and RDS, PostgreSQL database. They'd have to create all these different S3 buckets. They'd have to go run, create a Redis cluster, maybe through ElastiCache and set up all these security groups to open up all these rules, security group rules, get it all working.

All maybe using cloud formation, a bit of cloud formation, maybe using a bit of ansible, maybe using – calling CLI in AWS. Then the piece that's in Kubernetes is fairly standardized, because it's Kubernetes. This Kubernetes API.

One of the things that Crossplane does that helps with that problem is that we're able to do – deploy GitLab on the different cloud providers using just the Kubernetes API. That's the piece that we showed off yesterday with GitLab and do it in a way, in this case that is portable across clouds.

[0:18:38.0] JM: Okay. The thing is a lot of people probably have only interacted with git to the extent that they are reading and writing git commits to their local repository, or to repository on github. What's actually going on there is you've got a complicated distributed system that's hosted in github, where they're doing things to make your git repository reliable, easily accessible, the API layer is scalable and so on and so on.

When you decide I want to go with GitLab and I want to manage this thing myself and I want to be in the GitLab ecosystem, rather than the github ecosystem, if you want to self-manage it, you're going to have to spin it up on a cloud, or on your own servers. Assuming you want to spin it up on a cloud, you have to be the one who sets up that complicated distributed system. You have to be the one who sets up the metadata store in the MySQL database or PostgreSQL database, or whatever database it is, that describes the schema for your github repository.

The reason this makes it such a such a good example for what you're doing with Crossplane is in order to deploy and host a distributed system that is as complicated as GitLab, you have to go through a bunch of steps, you have to go through a big documentation plus tutorial thing, that's despite the fact that GitLab is a unicorn company that has every interest to make it as easy as possible to deploy GitLab, the point is that there is no way to make it easy to deploy. That is the problem you are solving.

[0:20:25.2] BT: That's one of the core pieces of Crossplane, which is how do we – the realization was we as an industry have now arrived at a standard API for running containers and essentially, the volumes that they consume. It's a standard API in the sense that I can go anywhere across every cloud provider and across on-premise environments. As long as everything fits in containers and volumes, I'm able to do that in a uniform, consistent, standardized way across clouds.

When it comes to things like what GitLab needs, which is Redis, buckets, PostgreSQL and all these other things, the choices that you have is either run all of those on a Kubernetes cluster, complex systems like PostgreSQL and Redis and that you'd have to now manage yourself. Or you have to leave the standardized API Kubernetes and use something completely different outside of it.

What Crossplane is doing is saying, “Hey, how about you manage all this infrastructure, including your cloud provider offers from the Kubernetes API?” Why don't we push the standardization line further beyond containers and beyond volumes and have it include things, like PostgreSQL, object storage, Redis, MySQL? How about even clusters themselves; EKS, GKE, AKS? Why don’t we make those all standardized APIs, such that you can take something like GitLab and the way you deploy it becomes kubectl apply GitLab? That's the only command you run to bring up production-ready GitLab on the different cloud environments that you want to run on.

[SPONSOR MESSAGE]

[0:22:18.7] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens and we don't like doing whiteboard problems and working on tedious take-home projects. Everyone knows the software hiring process is not perfect, but what's the alternative? Triplebyte is the alternative. Triplebyte is a platform for finding a great software job faster.

Triplebyte works with 400-plus tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz. After the quiz, you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple on-site interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte, because you used the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more, since those multiple on-site interviews would put you in a great position to potentially get multiple offers. Then you could figure out what your salary actually should be.

Triplebyte does not look at candidates' backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. I'm a huge fan of that aspect of their model. This means that they work with lots of people from non-traditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple on-site interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out. Thank you to Triplebyte.

[INTERVIEW CONTINUED]

[0:24:39.6] JM: I was talking yesterday to somebody about the operators, the Kubernetes operators. I've done a show on helm. You mentioned helm charts a little bit earlier. Helm charts are a package manager, maybe you could say.

[0:24:53.9] BT: It's three pieces. It's a templating language for llamo, a package manager and a lifecycle manager for lifecycle management of essentially, how to install, upgrade applications.

[0:25:07.0] JM: Okay. Helm charts let me deploy and install distributed systems to a Kubernetes cluster. The operator pattern, or the operator system gives me a way of deploying higher-level complex distributed systems that may involve helm charts. I think of operator pattern as a more declarative helm, maybe?

[0:25:36.1] BT: I would say that what the operator pattern provides is the ability to run custom operations logic, a controller, a custom controller of some sort, that follows the Kubernetes style reconciliation, where you're reconciling desired state and active state. You don't necessarily do that in helm, but I mean, you can install operators through helm, which makes this even more confusing. When you write an operator, you're basically adding new declarative logic, right? That's reconciling declarative state with the observed actual state.

[0:26:16.7] JM: You might be describing, okay I want a Cassandra operator, because I want a Cassandra cluster. Cassandra's hard to operate. Hey Kubernetes, give me the storage and etc., stuff that I need to run a Cassandra cluster. Why can't I just use the operator pattern to spin up GitLab?

[0:26:40.7] BT: That's a very good question. Let's break it up. Before we get to that, let's talk about Cassandra as an example. The choices are you could use – if you're in a cloud provider that has fully managed Cassandra service, right? Or maybe using data stacks as I mean, a fully managed service offering, that's – you basically consume that through an API. You say, “I want a Cassandra cluster.” You can set configuration on it through an API. It doesn't matter how it's running. It's the software vendor, the provider’s choice of technology for running it. They're giving you an SLA and they're telling you you can consume that and they're responsible for scaling it, backing it up and dealing with failures and upgrades and everything else, right? That's one deployment option, a fully managed service, managed service offering, right?

The next level from that is that I want to run it myself. I want to run Cassandra myself, whether it's on Kubernetes, or somewhere else. If you're running it on Kubernetes, you have two choices; you could either say, “I'm going to just run it as a stateful set that I deploy through a helmet, or however I deploy it.” You're using the basic primitives of Kubernetes to manage the Cassandra deployment. Or you’re using an operator. The idea of the operator is that it's actually automating some of the tasks that you would have to have done yourself, whether it's the scaling of a Cassandra cluster, or it's the upgrade, or dealing with all the consistency issues while you're upgrading and fault-tolerance around that. There is a fair amount of automation that you can do around that, but it doesn't get you all the way to a fully managed service and it certainly doesn't give you an SLA.

There's this essentially spectrum about automation, starting with very little automation, I'm going to run it myself in stateful sets, to operators, all the way to fully managed services, right? We just talked about Rook as a project, has a Cassandra operator. As one of the things that we orchestrate in Rook is a Cassandra cluster, right? To your question back to GitLab, GitLab consumes PostgreSQL databases and consumes Redis.

Should somebody deploy in GitLab in a cloud provider, should they use an operator to run the PostgreSQL database that GitLab consumes? Should they, or should they use a fully managed service, right? I think GitLab would tell you, and in fact, you go to their documentation, they highly recommend that you use the fully managed services of the cloud provider.

If you do that, you're stepping outside of the Kubernetes world and you have to do something very different. What we've done is we brought all of those under the Kubernetes API. If you wanted to use an operator, you can. We created an abstraction in Crossplane called Redis. It's a Redis plane. If you ask your applications built and your application says, “I need a Redis cluster.” That Redis claim could be served by an operator running Redis in cluster, or a fully managed service that is met in your cloud provider and it serves the same claim.

[0:30:01.7] JM: Claim. Redis claim. Redis claim as persistent volume claim.

[0:30:05.0] BT: It’s not. It’s actually a new type of –

[0:30:07.5] JM: It’s a new type of claim.

[0:30:09.3] BT: Yeah. We added all these different claims following the pattern of –

[0:30:12.8] JM: What is a claim? Define that term, claim.

[0:30:15.6] BT: It’s essentially a request to provision or use some piece of infrastructure, or resource. It acts as a claim for it. You can track what it’s bound to. When you say, “I need a Redis claim,” you're basically from a developer, or an application owner perspective describing your request for Redis. I need it to be a certain version. I needed to have certain properties on it. You can define all of that in part as part of the claim.

[0:30:45.8] JM: The idea is basically, you have a container that's anchored somewhere. It's anchored, let's say an AWS. If you want to attach storage to that container, that container needs to establish, or create, or instantiate a claim. The claim says, “I need storage somewhere, or I need resource somewhere.”

[0:31:14.0] BT: That's right.

[0:31:14.9] JM: Maybe it's a persistent volume claim, where you're saying, I need persistent storage. Maybe it's a Redis claim where it says, I need a Redis store. A Redis store has different persistence properties more broadly than persistent volume claim. Maybe you say, I have a queueing claim. I need a queue. You could imagine a range of specificity, inheritance, hierarchies that could be applied to a claim. The usefulness of the claim is it saying, "Here, I need this thing. Connect to it somehow." The thing that you are connecting to the claim, whether it's the persistent volume, or Redis, or whatever other queue, something you need to spin up to connect to that claim, can sit on another cloud provider, it can sit somewhere else.

[0:32:04.1] BT: Or be an operator in a cluster, or be, right? You're creating essentially a layer of indirection, like all problems computer science can be solved by an indirection. You're creating a layer of indirection where you've said from the developers perspective, they're going to define their requirements in terms of setting these claims. That claim could be bound to different implementations of it, depending on the environment, or depending on the policy that an administrator wants to set.

If you're running on a cloud provider and they have a Redis fully managed service, we bind the claim to it. If you are running on-premise, or on-premises and you don't have that choice, you should probably run an operator for Redis and the claim is bound to that. In both those cases, the application does not change. It's the infrastructure, or the admin perspective that does change. The model is very similar to persistent volume claims and storage classes in Kubernetes today, but it's now goes beyond just volumes. It goes to Redis and PostgreSQL and all the different things that typical modern distributed applications require.

[0:33:26.5] JM: One reason I think this is significant is if you think about the open source applications that have taken off, they're mostly developer tools. The consumer-facing open source tools, it's like WordPress. You've got WordPress, you've got how much else? It's WordPress, that's it. There are a lot of applications that you can – Why don't we have an open source TurboTax, right? Because nobody knows how to deploy that thing. It's not anybody's really interest to – I mean, it wouldn't make sense. We should have an open source TurboTax. Why not?

Probably, part of the impediment is the person who writes the open source code, who would hypothetically write it would be like, “Oh, my God. Setting this thing up to be deployed easily would be impossible. Therefore, I'm going to be managing it myself and open source doesn't actually matter. Therefore, why am I even doing open source? I'm just going to be a proprietary service provider.”

[0:34:34.6] BT: Think about what you just said in the context of what we saw with Kubernetes in the last few years, right? Most applications today have some Kubernetes deployment. The reason that that became easier is because we've standardized on how to run containers and their volumes, right? The logical question becomes why don't we standardize on more than just containers and volumes? Why don't we standardize on PostgreSQL and MySQL and Redis and other components that are critical and highly used from most applications?

If we were able to push the standardization line even higher, then as you said, deploying and running infrastructure becomes easier as a whole. It becomes easier to do that. If we can do that in a way without changing the managed service and deployment model of clouds and having people give you an SLA, that's actually even better, right? Now we don't have to – we're not pushing problem from one side to another. We're actually using things as they were intended to be used, but still standardizing on how to consume them.

[0:35:54.0] JM: As you're working with GitLab, is it giving you any insights as to what your business model is going to be for Upbound?

[0:36:01.8] BT: Yeah. I'm not sure we're ready to talk about the exact business model around this. What's interesting about GitLab and having GitLab be deployed through Crossplane in a portable way across clouds is it makes this just all the more real.

[0:36:17.8] JM: Yeah, it’s a great use case.

[0:36:18.7] BT: It’s a great use case. It's like having a real-world application. GitLab is an amazing product, but it's also – it's complex. There are lots of different moving parts to bring up something like GitLab. The cool thing is working with them is that getting that to be here is my

configuration for running GitLab and it's written in a way that consumes all these claims. Then somehow, the magic of deploying and running it in clouds happens in a standardized way. Seeing that come up end-to-end in the different cloud providers is really brings a lot of validation to the approach. That's the part that we announced yesterday and we're excited about.

You can imagine Upbound being in the business of making that available to enterprises, so that they could write their services and applications and reuse them and have a place for them to be able to run and deploy and manage them across different cloud providers. That's the path that we get excited about from a business standpoint, but it's still pretty early for us.

[0:37:29.6] JM: It's maybe something like – you don't have to talk about this. You don't have to tell me if I'm right. You could imagine the Upbound configuration marketplace, or something where GitLab strikes a deal with Upbound. In exchange, Upbound has a marketplace where basically, you go into Upbound, Upbound is your company for listeners who didn't know, and the user can say, "I want to deploy GitLab in a way that has my Redis on Amazon, my MySQL on Google, my queue on DigitalOcean, whatever." Maybe you as Crossplane get a cut of that, or maybe – or you could do consulting, or something. You got a lot of interesting business models you could go towards. I guess –

[0:38:19.7] BT: The one thing that we're doing that we feel strongly about is that that whole path, including the control plane and provisioning and picking of in your example, which were to deploy things, that whole path is open and it's community-driven. If there are companies like ours that will have – will be commercial – providing commercial products around it, we feel that that's more about a hosting a managed service, or a SaaS offering around it and still enabling everybody else to do the same if they wanted to. You can't arrive at an open plan without the entire path of provisioning and deployment being open.

[0:38:59.1] JM: What if you had a marketplace? Would you make the marketplace open source?

[0:39:03.1] BT: We'd be open to – Yeah, we would think about that.

[0:39:06.5] JM: That's very committal. I'm being sarcastic. It's very non-committal. I mean, no, no. Seriously, this is –

[0:39:13.7] BT: I mean, I would tell you about it, but we're early I would say.

[0:39:16.5] JM: Fair enough. Fair enough. I don't blame you. This is one thing like, you heard my conversation with Eric Brewer, where I was interrogating the idea of the open cloud a little bit, because not that I'm opposed to it. I'm certainly a fan of whatever – I mean, I'm certainly a fan of the cloud becoming more open than it is in the days of AWS total dominance, right? It is a question that I think people like you and what more immediately Google, are going to have to reconcile with okay, if we're open, does that mean that we need to fall under a certain set of licenses, or does that mean that we need to open source our marketing strategy? Does it mean we need to open source like what stuff Istio is doing to pull the puppet strings of the CNCF?

Maybe I'm just being a conspiracy theorist here. Maybe none of this matters. Maybe I'm totally wrong. I mean, it seems like there are some open questions to what it means to be an open cloud.

[0:40:23.2] BT: I completely agree. I think if they look at some of the things that we've seen in the last few years, certainly people, or cloud providers are willing to invest in open runtimes. If you look at a Kubernetes, you could describe it as a runtime for running containers, right? It is open. It's community-driven and it's have a healthy community around it and an open governance model and that's great. All the things around hosting Kubernetes and cluster management of it –

[0:40:54.2] JM: Right. GKE.

[0:40:55.6] BT: - and policy management.

[0:40:57.7] JM: IM privileges.

[0:40:59.5] BT: Everything else around it is all closed source and proprietary, right? It's a logical question to ask is like, is that an open cloud, right? Or should the definition of an open cloud go

beyond compute runtimes and also include all the cluster management and policy management, configuration management, the control plane, the API around that, the authentication and policies around it and everything else? What about should we go all the way to the cluster management software for things like MySQL and PostgreSQL that are behind the managed services, right? Where do we stop? At what point do you say we've opened up enough?

What you see right now is that if you're running a business where you're spending 10 billion dollars in CAPEX spend to build data centers, you have to have something, a big part of it that's actually you can lead with open, but there's a big part of it that's actually closed, because how do you differentiate from the other cloud providers?

[0:42:04.8] JM: I don't know, man. Here's the thing, is like, even if Google was to open source GKE and publish the spec for the TPU, or I don't know if it's already published, and open source BigQuery, who is going to run that? Are you afraid of DigitalOcean stealing your GKE technology, stealing your Tensorflow managed service technology? That seems paranoid.

[0:42:30.7] BT: On the flip side, you can ask the question when I do pay for whether it's Anthos, or GKU, or others, what am I paying for?

[0:42:39.1] JM: The data centers?

[0:42:39.8] BT: That's right. It's also the management in this experience around somebody else doing the security patches and updating it.

[0:42:47.0] JM: Yeah, those two.

[0:42:48.3] BT: A bunch of those are actually automated in software too.

[0:42:53.4] JM: Okay, sure. Open sourcing it doesn't mean it's easy to install and run.

[0:42:59.1] BT: Yup.

[0:43:00.6] JM: I mean, honestly to me, the biggest risk seems like, when you open source this stuff, you are – if there's security holes, I think that's the biggest risk. If everything becomes open, there's going to be too much open source software and there's not going to be enough eyes to catch the security flaws, until the people who have an incentive to catch it and catch it and then start mining Bitcoin on your infrastructure, that's the real risk from my perspective.

[0:43:28.9] BT: Yeah. If you take as a customer, or an enterprise customer, right? One of the things that why open source is interesting is am I betting, am I making architectural bets that for the next five years on technology, that essentially limits me from having other choices, or ties me to a single vendor, or all those things, right?

Open source is emerging as a way to reduce those effects. If I make a decision to use MySQL, I know I can get MySQL from a number of different providers, including on-premise and all those different places. If I use something like DynamoDB, that's available in exactly one place, right? You could apply that logic to all the different things that are part of your stack. If I use Kubernetes, I know I can get Kubernetes everywhere now, right? If I use IM and I create all these policies around IM, we get still using Kubernetes, does that limit my choice?

[0:44:34.8] JM: Yes.

[0:44:36.4] BT: Right. You're like, “Okay, so what's the definition of – how do you get more open? Is that important to the customers? Is it not?”

[0:44:48.1] JM: It’s in the best interest of the cloud providers. Because what we've seen is – what I have found inspiring about the seeing the KubeCons is like, as the community has become more open, the wallets of the enterprises have become more open. It's shown us that these enterprises are not stupid. They're making purchasing decisions, because things are more open. It feels like, okay, actually maybe they have been gun-shy about AWS for a while, because it's closed and it looks like an Oracle database basically and enterprises are not stupid. They got burned. They're not going to get burned again. Even AWS, you should become more open, because the enterprises are going to open their wallets more.

[0:45:40.4] BT: Right. If you do that, one, I'd say it's hard to do right now, because they've built a ton of proprietary stuff that would be really hard to decouple for everything else.

[0:45:54.0] JM: Yes, sure. Naturally. It's like the Borg problem.

[0:45:57.1] BT: It is the Borg problem. That's exactly right. I mean, even a lot of the things that Google is building today, even GKE and others are tied to Borg and everything else, right? You think about that and it's like, okay, well how do we – should we re-architect and open up our control planes, or do we continue making business as usual?

At some level, I feel part of the motivation of something like Crossplane is to say, "Hey, here's an idea. Why don't we start a community effort around a multi-cloud control plane? Maybe that could be the basis of open up more of the cloud surface area than just the runtimes."

[0:46:37.9] JM: How tricky would it be for you to do – I mean, I'm sure you thought about this, but how tricky would it be to do the hybrid, plus multi-cloud control plane through Crossplane?

[0:46:48.2] BT: I think it's a really interesting scenario. I think the idea that we were talking about of having claims that can be bound to different implementations of them – actually, we just showed a demo. We came out of a talk just before this at KubeCon, where there was a PostgreSQL claim being fulfilled bound to a CockroachDB cluster, which speaks PostgreSQL on the wire, orchestrated by Rook on-premise.
You basically have an application says, "I want PostgreSQL," so it specifies a claim for PostgreSQL. If you're running in AWS, you get that through RDS. If you're running in Google, you get that from a cloud SQL. If you're running on-premise, you get it through CockroachDB cluster running orchestrated with work on-premise. That's the application does not care.

[0:47:43.3] JM: Why would they do that? I mean, I know CockroachDB gives you more availability, or not availability. Global consistency, right?

[0:47:54.3] BT: That's right. In this case, we're using Cockroach as if it was a PostgreSQL database, because that's what was available for this demo. The nice thing about Cockroach is it does speak PostgreSQL on the wire. It's speaks the –

[0:48:05.5] JM: I just don't understand why they didn't just use PostgreSQL, or –

[0:48:08.4] BT: Oh, because it actually has better scale properties and it has – it's easier to manage a classic PostgreSQL database. You could actually drop and replace PostgreSQL with a Cockroach cluster.

[0:48:20.4] JM: Changing the topic, so you listen to that Eric Brewer episode and that long 30-minute preamble I did that talked about the intricacies of the different clouds. What was your perspective on that? Was there anything you thought I got wrong about the dynamics of the cloud provider wars?

[0:48:43.2] BT: I think you should do more preambles like this, the long ones. That was my perspective on it. I thought it was good. I thought that there's definitely really interesting dynamics happening right now between the different cloud providers.

[0:48:55.9] JM: It's Game of Thrones.

[0:48:57.2] BT: The open-source community in general I'd say.

[0:48:58.6] JM: It's so political.

[0:48:59.7] BT: It is very political. It's not surprising. I mean, they're all in. I mean, it's a really interesting – Part of why I'm excited about being in the space right now is I feel the open source community will needs to be more involved in this game.

[SPONSOR MESSAGE]

[0:49:28.9] JM: Commercial open source software businesses build their business model around an open source software project. Software businesses built around open source software operate differently than those built around proprietary software.

The Open Core Summit is a conference for commercial open source software. If you are building a business around open source software, check out the Open Core Summit, September 19th and 20th at The Palace of Fine Arts in San Francisco. Go to opencoresummit.com to register.

At Open Core Summit, we'll discuss the engineering, business strategy and investment landscape of commercial open source software businesses. Speakers will include people from HashiCorp, GitLab, Confluent, MongoDB and Docker. I will be emceeing the event and I'm hoping to do some onstage podcast-styled dialogues.

I am excited about the Open Core Summit, because open source software is the future. Most businesses don't gain that much by having their software be proprietary. As it becomes easier to build secure software, there will be even fewer reasons not to open source your code.

I love commercial open source businesses, because there are so many interesting technical problems. You've got governance issues. You got a strange business model. I'm looking forward to exploring these curiosities at the Open Core Summit and I hope to see you there. If you want to attend, check out opencoresummit.com. The conference is September 19th and 20th in San Francisco.

Open source is changing the world of software and it's changing the world that we live in. Check out the Open Core Summit by going to opencoresummit.com.

[INTERVIEW CONTINUED]

[0:51:30.8] JM: We were initially introduced by Joseph Jacks from OSS Capital. When he started that venture firm, I thought, I was like, okay, nice to marketing strategy for your infrastructure venture capital firm. Okay, it's nice. Cool. You only invest in open source companies. Big deal. I think he saw something ahead of time that open source really is transforming the software industry. Because it is so transformative, there are some dramatic changes that occur in the diplomacy of different software companies with each other of the go-to market strategies of these different companies, of the herding of cats, of all the different software development communities. There are so many changing dynamics.

[0:52:28.8] BT: It's a movement. It goes beyond just an Apache license on code repository for sure.

[0:52:38.5] JM: The politics, it's not just the major cloud providers. You also have the burgeoning companies, like you, or GitLab, or Buoyant. You have some dark horses, like Mesosphere or DigitalOcean. As one of these companies that is deep in this political space, do you have any suggestions for companies that are on the upstart and they're starting to see that okay, there are all these different relationships and alliances and collaborations. What are your general principles for intercompany dynamics?

[0:53:22.3] BT: One suggestion I would have is that try to get traction with the community first. Get folks that are engineers, DevOps, developers in general using your projects, giving you feedback, get traction with the community. When you get traction with the community, it's hard to ignore. You could use that to help with some of these relationships and dynamics that you were talking about. If you're a small company, you're basically trying to navigate through all the different political issues and business issues without a community backing, it's a lot harder.

[0:54:07.5] JM: What is unique about building a company in this space that you could tell me that I'm unlikely to hear from anyone else?

[0:54:18.7] BT: I'd say that it's really easy to be dissuaded from entering a space like this, or thinking about is this even possible? I would argue that this cycle of disruption, or going from things that are proprietary openness have happened many times. I feel if you have a team that's passionate about delivering something in the space, that it's likely – you're likely to actually make it happen.

[0:54:47.7] JM: How did you manage to – I don't know how you managed to raise money and then build a team around this idea, because when you started, when you just had Rook, you were so far from having anything concrete in place around actually making a multi-cloud thing happen.

[0:55:07.5] BT: I think we had – I mean, we're still early. I can tell you that we are –

[0:55:12.2] JM: Do you ever feel dissuaded, even today?

[0:55:15.0] BT: As a startup founder, you go through the ebbs and flows of being dissuaded and optimistic every minute. Obviously, I feel very passionate about this problem and the team around Upbound and the team around all these – the community around each of our projects, I feel is passionate about the space. As a result, I mean, we hope to get more people around it and I think things like this happen if you, or can get the right set of folks that are passionate about them and they just – they happen.

[0:55:54.8] JM: Do you ever reflect on the question of fake it till you make it? Because you see sometimes this this works out well. I think your company is a case of when I met you two years ago and you were talking about this, it was like a cool vision. Clearly, you're a smart guy. I look you in the eyes and it seems like you sincerely believe that this is something that's going to exist, whether you build it or not, and you can convince me of that.

Then you see something like Theranos and it's like, “Wow, huh. Fake it till you make it has a cost.” Or you see something like Istio and it's – not to compare Istio to Theranos, or to compare you to Istio, or to compare you to Theranos, but there is this dynamic where you lay out a vision, the vision is important, you get some money and then you market that vision. Sometimes it works out and sometimes it ends up looking like snake oil. The only people who end up writing the historical story are the winners.

[0:57:02.5] BT: That's right. I would say though that even when you look at large companies that are pushing new initiatives and potentially new infrastructure and new platforms, I'd say, would you not say that they're also fake it until you make it?

[0:57:18.2] JM: Absolutely.
[0:57:19.3] BT: That happens outside of start – it happens in big companies as well, right? I think that you have to have a deep belief and a vision around what needs to happen and you have to hit it hard. If you're right about it and you are able to execute against that and you got to tell great stories about it. If you don't, then it doesn't happen.

[0:57:44.1] JM: Unless, the Wall Street Journal writes a great story about it. Wall Street Journal has not written about Istio. Are the service mesh wars winner-take-all?

[0:57:53.2] BT: No, I don't think so. I think it was interesting to hear about SMI yesterday.

[0:58:01.7] JM: Does that seem like a good abstraction to you? Explain what SMI is, to the extent that you understand it.

[0:58:06.5] BT: It's essentially an interface for service meshes, so that you could plug in different service meshes as I understood it. I've spent two minutes on it. I mean, I think that I don't think it's a winner take all. I think it's a really interesting thing that's happening, but I don't see it as a winner-takes-all.

[0:58:27.2] JM: Here's the thing, your service mesh is going to need a bunch of integrations, correct?

[0:58:33.0] BT: Mm-hmm.

[0:58:34.0] JM: Those integrations are going to be written by ISPs, independent software vendors. They are not going to want to write multiple integrations. They're not going to want to write an integration for Istio and an integration for Linkerd. That to me seems –

[0:58:51.9] BT: Doesn't SMI essentially help with that?

[0:58:54.8] JM: Yeah, maybe, but wouldn't it just be easier if we just had one service mesh? I feel we went through this with container orchestration wars. This was the same thing. You can remember from KubeCon, or DockerCon whatever, 2015-2016, we support Docker Swarm and Mesosphere and Kubernetes and –

[0:59:13.7] BT: We ended up with CRI.

[0:59:17.0] JM: Well, yeah. Okay, I guess. I mean, tell me more about CRI. How does CRI map to that historical –

[0:59:22.2] BT: You can plug in your container runtime behind the CRI interface and whether it's Docker runtime, or rockets, or all those different runtimes.

[0:59:34.3] JM: What's the most popular runtime? Is it the Docker runtime?

[0:59:37.5] BT: The Container D one.

[0:59:38.7] JM: Container D. That's the one that displaced Docker.

[0:59:42.6] BT: Well, it's the same one that's used by Docker.

[0:59:44.4] JM: Oh, the same one used by Docker. Well, then that one won, right? Then you have an –

[0:59:48.3] BT: It sits behind essentially an interface CRI right now. Kubernetes consumes it through CRI as I understand it. You could look at SMI and say, “Here's an interface to consume service meshes, or use service meshes.” There might still be a dominant service mesh behind it. Now you could argue standardized. If there are, to your point of all these folks that are doing integrations, if they do integrate at the SMI line, then maybe the problem gets easier for all of them too.

[1:00:24.2] JM: Yeah. Yeah. I mean, I guess it's –

[1:00:26.0] BT: It’s early. Again, this is –

[1:00:28.8] JM: Speculation.

[1:00:29.9] BT: This is pattern matching on –

[1:00:31.1] JM: It’s pattern matching, which is all I’m doing. That’s all I’m – Yeah, fair enough. Okay. You think I'm too hard on Istio?

[1:00:39.8] BT: Not really.

[1:00:41.5] JM: Why did you raise money from Google Ventures? Did you think about going with Sequoia, or Andreessen Horowitz, or any of the other major venture capital firms?

[1:00:50.1] BT: I think you picked – if there's a lot of VC attention, you got to pick the one that you think is the best fit at that time. We thought GV was the best fit for us at the series A. We really like them. They've been super helpful. That's why we went with them.

[1:01:09.0] JM: Last question. It's mid-2019, what are the competitive dynamics between Google Cloud and AWS?

[1:01:21.1] BT: Well, AWS has the dominant market share. I'd say that if you look at what Google's done, if I understand it correctly, is that they essentially went after a more open source model. Why Kubernetes was open sourced to essentially level the playing field around compute and containers. That was quite successful.

Amazon now supports Kubernetes, so you could argue that whatever move happened with Kubernetes as a threat has somewhat been neutralized now, in the sense that yeah, I mean, you can get – you can go run on EKS. I can tell you from a tech standpoint, GKE is a lot more usable than EKS. You still get Kubernetes in both places and you can run your stuff there. If you think the dynamic, I'd argue that Google has more – has to do more plays to try to get more market share. Taking the more open cloud, I think what they've done with working with companies like Elastic and Mongo and others to bring their managed services into Google's cloud offering are really interesting moves. Whereas, Amazon is taking the – we're going to protect our supply chain of open source and we'll do whatever is needed to keep the open source supply chain.

[1:02:55.2] JM: What do you think of that whole licensing debate? Elasticsearch getting angry at Google, Redis getting angry at Google.

[1:03:04.5] BT: Amazon, you mean.

[1:03:05.5] JM: Right. Amazon, sorry.

[1:03:08.0] BT: I mean, it's an interesting dynamic. It's Amazon –

[1:03:10.6] JM: Did Amazon do anything wrong? That was I took issue with, is they – the Elasticsearch, for example, the blog post in Elasticsearch, I think the CEO, or the founder was very indicting of AWS. I was like, "It's open source. There's no rules."

[1:03:29.2] BT: Yeah. I mean, I think if you step back and look at that dynamic, you could argue that Amazon is saying that the open source community is welcome to have a business around open core, selling commercial software on top of open source and they should do that and they should do it cleanly. You can make business on, you can make professional support on, you can do all the stuff. When it comes to a SaaS business, that they would like to do it on their platform and make sure it's protected from that perspective.

I often wonder what happened in the – why didn't Amazon, for example, make a deal, a biz dev deal to bring in one of these commercial open source companies and have them run as a managed service on Amazon platform. Was it a failure on agreeing on a price, or was it even attempted?

[1:04:25.7] JM: I thought if you bought Elasticsearch from Elastic on the Amazon marketplace, that is the deal, that is the biz dev deal.

[1:04:34.9] BT: Well, let's take that case specifically. Amazon has an Elasticsearch service. They'd had one since I think 2015, right? It's based on the upstream version of –

[1:04:45.4] JM: It's called Amazon Elasticsearch Service.

[1:04:47.9] BT: Correct. It's based on the upstream version of Elasticsearch. Elastic NV, the company has a downstream version with commercial modules and all sorts of differentiated features that they've built, right? If you wanted to use some of those commercial features, you actually have to buy a license from Elastic, or use their managed service offering. It wasn't available on the Amazon platform from Amazon. You could buy the managed service offerings from Elastic NV, the company.

[1:05:19.4] JM: You couldn't buy it on the Amazon marketplace?

[1:05:21.9] BT: Well, no, you can go through them and they would deploy it in Amazon, even within the same VPC, but you're not – at that point –

[1:05:29.7] JM: You can talk to Elastic and say, “Hey Elastic, spin me up a cluster,” they spin it up for you.

[1:05:33.6] BT: They spin it up for you and you can put it in the same VPC, and so you can connect to it. You don't get IM integration, you don't get cloud watch, you don't get – it's a separate world.

[1:05:41.9] JM: Wait, why can't you – You can't do that through the marketplace, the Amazon marketplace?

[1:05:45.7] BT: In a marketplace, you essentially deploy AMIs on your own VPC.

[1:05:50.0] JM: Oh, that’s Amazon Machine Instances.

[1:05:51.0] BT: That's right. You deploy Amazon Machine Instances and you run them. If you want to use – if you want somebody to actually run a managed service offering that they're doing the backups, they're doing the whole thing, you have to go through their manage – a managed service offering. You're either running Amazon's managed service and using the upstream version of, or you leaving the walled gardens of Amazon, you're leaving the IM integration, cloud watch integration, everything else and going to Elastic offering, which has a ton of features, but is not as integrated into the Amazon platform as they'd like to be.

Your choice becomes, do I use a good enough Elastic service and it's fully integrated? It's one API click away, it's integrated with IM, it’s cloud watch, everything, cloud formation, the whole thing is integrated.

[1:06:45.1] JM: The analogy here is if you use an iPhone, Siri always tries to get you to use Apple maps. You're like, “Siri, stop it. Let me use Google Maps.” I'm sorry. I am giving you the fully integrated offering.

[1:06:59.6] BT: You get the fully integrated experience.

[1:07:01.0] JM: That is worse.

[1:07:01.6] BT: That's right.

[1:07:02.2] JM: It's integrated, but it's worse, which is what Amazon Elasticsearch service is relative to the one that you buy from Elastic, because for Elastic it’s existential. They button up the buttons and zip up the zippers and it's much, much better, is my understanding. It's not just incrementally better, it's significantly better.

[1:07:20.4] BT: It’s better, but it’s not as integrated –

[1:07:22.2] JM: Not as integrated.

[1:07:23.9] BT: It's basically becomes a trade-off between deep integration and good enough, or best of breed, but not integrated, isolated, fragmented, right? That dynamic is really what's at play here. When Elastic the company relicensed and started saying that we’re guarding against hosting of our service, Amazon could have taken one of two choices. They could have said, “Okay, let's go work with them and bring the best of breed offering and make it integrated into Amazon,” right? That requires biz dev, that requires a relationship. I'm sure, I can – I would bet money that it was attempted, but they couldn't agree. As a result, Amazon reacted with, “Okay, well we're going to create a new distro for –”

[1:08:19.8] JM: Let's be honest, Elastic didn't have a whole lot of leverage in that conversation.

[1:08:23.1] BT: That's right. If you look at Google and this is what I’m saying, Google cloud was interesting here is that they took a different approach. Google cloud said, “You know what? We will work with you. We will bring Elastic with all the features and best of breed and we'll try to

integrate it as deeply as possible into the Google cloud platform." I believe Azure is doing the same. That's a different approach for partnership than Amazon has taken with at least in this case, Elastic.

[1:08:52.2] JM: It's a bright world for Crossplane.

[1:08:54.7] BT: We think so. Yeah.

[1:08:56.5] JM: Well Bassam, thanks for coming back on the show. It's been really fun talking.

[1:08:59.0] BT: Yeah, thank you. It's always a pleasure to be here.

[END OF INTERVIEW]

[1:09:05.4] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source, it's free to use and GoCD has all the features that you need for continuous delivery. You can model your deployment pipelines without installing any plugins. You can use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your cloud native project.

With GoCD on Kubernetes, you define your build workflow, you let GoCD provision and scale your infrastructure on the fly and GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, and they have talked in such detail about building the product in previous episodes of Software Engineering Daily. ThoughtWorks was very early to the continuous delivery trend and they know about continuous delivery as much as almost anybody in the industry.

It's great to always see continued progress on GoCD with new features, like Kubernetes integrations, so you know that you're investing in a continuous delivery tool that is built for the long-term. You can check it out for yourself at gocd.org/sedaily.

[END]