EPISODE 738

[INTRODUCTION]

[00:00:00] JM: Cloud providers created the ability for developers to easily deploy their applications to servers on remote data centers. In the early days of the cloud, most of the code the developer wrote for their application could run on any cloud provider, whether it was Amazon, or Google, or Microsoft. These cloud providers were giving developers the same Linux servers that they could expect from an on-premise deployment.

Early cloud applications, such as Netflix, and Airbnb, and Uber took advantage of this cloud infrastructure to quickly scale their businesses. In the process, these companies had to figure out how to manage open source distributed systems tool, such as Hadoop and Kafka. Cloud servers were easy to create, but orchestrating them together to build distributed systems was still very hard. As the cloud providers matured, they developed higher level systems that solved many of the painful infrastructure problems; managed databases, auto-scaling queuing systems, machine learning APIs, hundreds of other tools. Examples included Amazon Kinesis and Google BigQuery. These tools are invaluable, because they allow a developer to quickly build applications on top of durable, resilient cloud infrastructure.

With all of these managed services, developers are spending less time on infrastructure and more time on business logic, but managed services also lead to a new infrastructure problem. How do you manage resources across multiple clouds? A bucket storage system like Amazon S3 has different APIs than Google Cloud Storage. Google Cloud Pub/Sub has different APIs from Amazon Kinesis. Since different clouds have different APIs, developers have trouble connecting cloud resources together and it has become difficult to migrate your entire application from one cloud provider to another.

Crossplane is an open source control plane for managing resources across multiple clouds. The goal of Crossplane is to provide a single API service for interfacing with all the parts of your application regardless of what cloud they are on. Crossplane is a project that was started by Upbound, a company with the goal of making multi-cloud software development easier. Bassam

Tabbara is the CEO of Upbound and he joins the show to talk about multi-cloud deployments, Kubernetes Federation and his strategy for building a multi-cloud API.

We are running a survey for listeners. Please go to softwareengineeringdaily.com/survey to fill it out. We'd love to get your feedback. If you want to be entered in to win a piece of Software Engineering Daily swag, you can enter in your email address and fill out that survey to get a free hoodie, or t-shirt or something.

I also want to mention, you can sign up for our newsletter at softwareengineeringdaily.com/newsletter, and as always you can find all of our old episodes by downloading the Software Engineering Daily app for iOS or Android. You can check those out. With that, let's get on with the episode.

[SPONSOR MESSAGE]

[00:03:26] JM: For years, when I started building a new app, I would use MongoDB. Now I use MongoDB Atlas. MongoDB Atlas is the easiest way to use MongoDB in the cloud. It's never been easier to hit the ground running.

MongoDB Atlas is the only database as a service from the engineers who built MongoDB. The dashboard is simple and intuitive, but it provides all the functionality that you need. The customer service is staffed by people who can respond to your technical questions about Mongo.

With continuous backup, VPC peering, monitoring and security features, MongoDB Atlas gives you everything you need from MongoDB in an easy to use service, and you can forget about needing to patch your Mongo instances and keep it up to date, because Atlas automatically updates its version. Check out mongodb.com/sedaily to get started with MongoDB Atlas and get $10 credit for free.

Even if you're already running MongoDB in the cloud, Atlas makes migrating your deployment from another cloud service provider trivial with its live import feature. Get started with a free three-node replica set. No credit card is required.

As an exclusive offer for Software Engineering Daily listeners, use code SEDAILY for $10 credit when you're ready to scale up. Go to mongodb.com/sedaily to check it out, and thanks to MongoDB for being a repeat sponsor of Software Engineering Daily. It means a whole lot to us.

[INTERVIEW]

[00:05:24] JM: Bassam Tabbara, you are the founder and CEO of Upbound. Welcome back to Software Engineering Daily.

[00:05:29] BT: Glad to be here.

[00:05:31] JM: Your company, Upbound, is focused on supporting multi-cloud deployments, and the most common reason that people give, at least when I talk to people for wanting to be multi-cloud, is to avoid vendor lock-in. But there are actually plenty of other reasons that an enterprise might want to go multi-cloud. What are some of the reasons to go multi-cloud?

[00:05:54] BT: There're actually a number of reasons, and lock-in definitely is one of them. But if you kind of think about maybe the emotional way of thinking about it, there are actually good reasons to be multi-cloud. Now that there are a number of cloud providers, they have differentiated services. So if you are able to use their differentiated services, run across them, then there are benefits to that. Whether they all go through a different innovation cycle, whether it's using some of the instance types or using some of their AI or ML, managed services or other new managed services that they provide, if you're all in on one, you don't get to use the differentiated services off the other cloud provider.

There are others too. I'm happy to walk through them. But that one is pretty high on the list. Another one that comes to mind is some people don't get to decide which cloud providers they get to use. For example, if you're a large enterprise company and you acquire a smaller company and they're on the other cloud, you automatically become multi-cloud, or you have an on-premise presence, and people call that hybrid, but sometimes that's also – That is a form of being multi-cloud. There are lots of reasons that people become multi-cloud or are interested in becoming multi-cloud besides just, "I don't want to get locked-in."

[00:07:18] JM: So the managed service argument is the one that I see as the best argument for going “multi-cloud”, but it seems like if you’re building your software with as many managed services as possible, which I think in general you should be doing depending on what your product is. But for many products that are high-margin software products or startups that are just trying to make things work, and if it works, then you have a high-margin software product. Then you should pretty much go with these managed services because you just get more flexibility. You get to move faster. You get guaranteed uptime. But if you’re stitching together a system from, let’s say, Google BigQuery, and Amazon DynamoDB, and AWS Lambda, and let’s say you’re using Microsoft Azure. They have a Pub/Sub product. You’re using all these things together. That’s not really like you’re multi-cloud, right? That’s not the same as having a multi-cloud Kubernetes setup, for example, or having VMs running on these clouds. It’s more like you’re just buying. It’s almost like you’re buying compact disc software from these different companies.

[00:08:30] BT: Right. In general, I think organizations don’t want to run complex platform software. So using managed services makes a ton of sense, because then you get to focus on the things that help your business thrive and not how to backup database or deal with transaction and consistency and all sorts of other issues that platform software typically runs into.

So using managed services is great. What’s happening today is that most managed services are – When you think of managed services, you think of your cloud provider’s managed services, whether it’s managed MySQL, or managed PostgreS, SQL, or managed Redis, or all the things that say your cloud provider offers.

So if you’re interested in using a best of breed managed services, there are certainly the ones that your cloud providers offers, but there are also ones that are outside of your cloud providers. For example, if you wanted to run Spanner or a variant of Spanner and you’re in AWS, there isn’t one. So you might want to look in to, say, CockRoachDB, or if you wanted to run – They just added, for example, managed Kafka. That didn’t exist before, but there were alternatives from the companies that support these products.

So at some level, being multi-cloud and using multiple managed services doesn't mean that you should be restricted to the cloud offerings of your cloud provider, but instead you should actually use the best of breed managed services available to you that make your business or your application or your workloads work really well. Does that make sense?

[00:10:02] JM: It does. So if I'm running let's say a Ruby on Rails application on AWS server. So whether it's running on a Kubernetes cluster, in some Docker containers, or it's running on AWS VM, what is it that makes it easier to integrate with AWS services? Why is there any difference between if I'm just on these AWS servers, why is there any difference between how I integrate with, for example, AWS Kinesis or with Google BigQuery?

[00:10:34] BT: That's a very good point. So if you're using any of your cloud providers managed services, so your example here of Kinesis, or Dynamo, or RDS if you're in Amazon, all of those services are integrated at the API level and at the management console level and actually have features between the different services that offer a high-degree of integration, right?

For example, you could – Using the same AWS API, or the CLI, or the management console, you can provision Kinesis, Dynamo, Redis, all the managed services that are available to you as well as the VMs and Lambda and everything else available to you from Amazon. What you can't do is reach out through that same API which is – Obviously, behind the API is a control plane, and call a third-party's cloud offering. There is no easy way to do that.

You could run Amazon, for example, and other cloud providers offer a marketplace and there you could run third-party cloud offerings as VMs mostly, and the only integration you get there is billing integration. You don't get API integration. You don't get management console integration. You don't get integration across services. So they're essentially – The apps in the marketplace or the managed services in the marketplace are essentially third-class offerings within that cloud provider. It is very different when you're using a managed service offered by your cloud provider versus a managed service that is not offered by your cloud provider.

[00:12:15] JM: Do a difference would be if I spin up an instance on AWS, that instance comes with the Amazon CLI, the command line interface, and I can do things from that command line interface maybe like – What? Spinning up a Kinesis connection or something, but you can't – Okay. But I can't – There's no first-class access to Google BigQuery from this Amazon instance.

[00:12:43] BT: There is not. We'll ignore the networking egress cost and network connectivity across clouds for a second, but even if you compare, say, running in Amazon and you wanted to use one of Amazon's managed services, let's take Kafka. They just announced Kafka. So it's fresh in my mind. If you were to use the version that Amazon provides, you get complete integration into their platform. If you were to use managed service or cloud offering of Kafka that is not offered by Amazon, you get at most billing integration and nothing else.

[00:13:20] JM: I see.

[00:13:20] BT: So that's just the nature of how cloud providers work. So the interesting part there is that it becomes a tradeoff. Should I use my cloud provider's managed services and get the integration, but potentially only I'm limited by the features that they support or I'm limited by their innovation cycle on these managed services, or if it's an open source product, I'm limited by what is upstream and not open source product versus use a third-party, independent cloud offering, and I get very little integration at the API level or at the management console level. The third option is I get – Maybe I should run my own, and especially if it's open source. There it's no longer a managed service. The enterprise or the organization is running the software themselves. So those are the three that people kind of follow today.

[00:14:16] JM: So if you're a startup, like let's take a later stage startup, like Thumbtack. So I did a show a year and a half or so ago with Thumbtack and they are mostly on AWS mostly using ECS for managing Amazon's container service to manage their application, their node.js services, maybe some Java services, and they use some AWS services, but they also use Google cloud Pub/Sub. They use Google BigQuery. They use some other Google services. You probably don't know exactly what Thumbtack is doing, but when you see companies like that where most of their core infrastructure is on AWS, but they are using some Google cloud managed services, what is their way of connecting to those Google managed services even though their main application is on AWS?

[00:15:14] BT: It’s very likely just the internet if they’re connecting between clouds right now. There are some companies that offer ways to connect across clouds that are not just through public internet, but the most common one is just through the public internet.

[00:15:29] JM: When you say through the public internet, what does that mean in contrast to being on a cloud? Like all your services on the same cloud?

[00:15:37] BT: Most cloud providers distinguish between network traffic that goes in an out of the cloud provider and network traffic that is within a cloud provider. When you egress out of the cloud provider, you’re typically just using the backbone of the internet to connect, kind of like how say on your laptop connecting from a browser to a cloud provider. It’s all public traffic at that point. Those are typically taxed. So cloud providers love it when you bring data and network traffic into the cloud provider, but they tax you on any data leaving the cloud provider.

[00:16:14] JM: What are the other frictions for that kind of customer, the kind of startup like a Thumbtack that wants to have multi-cloud capabilities?

[00:16:26] BT: I mean the biggest one that we at Upbound think about a lot is the lack of a control plane that spans clouds. If you are all within one cloud provider, then you have a control plane that gives you a uniform API, a consistent API, and a set of managed services that are integrated into it and work well together. The minute you cross or leave or use a manage service outside of that realm, you’re back to doing things on your own, whether it’s tools that help you with provisioning, or scripts, or home-brewed scripts, and you’ll see a lot of people build a lot of those themselves, and there isn’t, if you will, an API in a controlled plane that is letting you do your work across the cloud providers, or even across cloud providers and independent cloud offerings. That doesn’t exist.

[00:17:24] JM: What do you mean by that term control plane?

[00:17:27] BT: In general, people use the word control plane, I think it comes from essentially network switches where they differentiate between traffic that is used to move bits, so a data plane, a plane where data moves. Then a control plane is where control signals are moved, but not data. Essentially, a fancy word for saying it's a control channel or a control – The ability to control things, but not be in the path of moving data, or connectivity, or anything like that.

[SPONSOR MESSAGE]

[00:18:09] JM: How do you know what it's like to use your product? You're the creator of your product. So it's very hard to put yourself in the shoes of the average user. You can talk to your users. You can also mine and analyze data, but really understanding that experience is hard. Trying to put yourself in the shoes of your user is hard.

FullStory allows you to record and reproduce real user experiences on your site. You can finally know your user's experience by seeing what they see on their side of the screen. FullStory is instant replay for your website. It's the power to support customers without the back and forth, to troubleshoot bugs in your software without guessing. It allows you drive product engagement by seeing literally what works and what doesn't for actual users on your site.

FullStory is offering a free one month trial at fullstory.com/sedaily for Software Engineering Daily listeners. This free trial doubles the regular 14-day trial available from fullstory.com. Go to fullstory.com/sedaily to get this one month trial and it allows you to test the search and session replay rom FullStory. You can also try out FullStory's mini integrations with Gira, Bugsnag, Trello, Intercom. It's a fully integrated system. FullStory's value will become clear the second that you find a user who failed to convert because of some obscure bug. You'll be able to see precisely what errors occurred as well as the stack traces, the browser configurations, the geo, the IP, other useful details that are necessary not only to fix the bug, but to scope how many other people were impacted by that bug.

Get to know your users with FullStory. Go to fullstory.com/sedaily to activate your free one month trial. Thank you to FullStory.

[INTERVIEW CONTINUED]

[00:20:31] JM: If we take the current model of a startup, like that Thumbtack example, where they've got their compute instances on AWS and they want to use managed service in Google

Cloud and they have to go across the internet and pay the tax of going across the internet and not have nice integrations built into their CLI. How does that contrast with the ideal world? In your world, what would be the ideal way that Thumbtack would be interacting with their own services and the managed services?

[00:21:11] BT: In our world we think of – Imagine a world where there was an API that enabled you to both provision managed services on Google and Amazon, and that API was served by a control plane that was able to provision and managed a full lifecycle of these services, deployed them, connect them to each other, and that was all possible essentially through an API endpoint that you get to call from your favorite tools, CLI, or even could have a UI in front of it. Today, these control planes have a scope of one cloud provider. They don't span cloud providers.

[00:21:57] JM: Got it. I know this is what you're working on Upbound, this cross-plane, control plane that allows you to access different cloud providers. Before we go there, I want to talk a little bit about Kubernetes, because we haven't talked much about Kubernetes. Kubernetes came out at a time when more companies were looking at multi-cloud, where they were looking at hybrid cloud, they were looking at adding cloud functionalities to their on-prem infrastructure. How does Kubernetes fit into – When you talk to these companies that have been thinking about going hybrid cloud, or multi-cloud, how does Kubernetes fit into their strategy?

[00:22:37] BT: I think, honestly, Kubernetes is probably the closest we've come to having a common and consistent API across cloud providers. We're obviously big fans of Kubernetes with another project that we started, the Rook project, is all based on Kubernetes and is one of the leading storage orchestrators on Kubernetes. So if you think about what Kubernetes offers, it gives you an API that lets you orchestrate the running of containers across a set of machines, nodes. These nodes could be VMs, or physical machines. It really doesn't matter.

But that API to orchestrate the running of these containers and some of the resources they use, specifically volumes, storage volumes, works consistently whether you are running in Amazon, running in Google, you're running on Azure, or you're running on-premise, on premises, right? That you have a consistent API that lets you do that, and that is really, really interesting, because it offers a layer of portability across cloud providers and essentially starts hitting on some of the points we were talking about earlier for why people want to be multi-cloud.

So what you see with Kubernetes, and it's caught a ton of momentum around it, with all the momentum, it really only knows how to orchestrate containers and volumes, that it's a really amazing system that does the orchestration of both of those things. But people run more than just containers and storage volumes connected to containers in the cloud. They were on databases. They were in message queues. They do Pub/Sub. They run other forms of workloads besides containers, whether it's serverless, or VMs, or other things that are running across the cloud providers.

For those, there really hasn't been any attempt at portability across the cloud providers just that Kubernetes has done for containers and volumes. So what we're seeing is that people have adapted Kubernetes primarily because it gives them the ability – Both Kubernetes and containers give them the ability to be a little more portable across their environments. But when they reach out and talk to a database, that promise of portability doesn't hold.

So what we're seeing is that people are starting to run databases using an operator pattern, like the ability to run software like MySQL or PostgreS or others. Rook is a good example of that too, running storage systems directly on top of Kubernetes, which you can argue, "Why would you want to do that? Why would you want to run your own database on Kubernetes?"

One of the reasons is that you get a form of portability for databases when you do that, but you're trading off using a managed service when you do that. You're back in charge of running and managing your own database and giving yourself an SLA versus using a managed service.

[00:25:38] JM: Right. So we talked about that in our last show. We did a pretty deep dive into Rook. If I remember, I should have reviewed that show before this one. But if I remember, you can use Rook to – Like Rook is like a storage interface. So you can use it to connect to Amazon's elastic block storage, or whatever Google's block storage is, right? So it's a common interface between the storage systems of different cloud providers.

[00:26:07] BT: Yeah. I mean, Kubernetes will let you today – Kubernetes has an abstraction called persistent volume that lets you use whatever block storage or file system storage that the cluster supports. So if you're in Amazon, you're likely using EBS. If you're in Google, you're using Google Persistent Disk. That abstraction, the persistent volume, is in the core of Kubernetes.

What Rook does is it lets you run the server side of this abstraction, the thing that provides storage. It lets you run the storage system itself on Kubernetes.

[00:26:44] JM: And that's like a file system?

[00:26:45] BT: Yeah. You can think of it as a file system, or block store, an object store, or there are numerous kinds of storage that offer a persistent volume to Kubernetes. That server side is with Rook and it's self-run on Kubernetes. In those scenarios, you're likely not wanting to use EBS, or Google Persistent Disk, or you want object storage to run on-premise, or you want block storage to run on-premise. So there are numerous scenarios where Rook makes sense in this story.

[00:27:18] JM: Could you give a few of those examples? Maybe just give an outline of the Rook project. The Rook project is an open source project that you have been working on for a while. This was early in Upbound's history. I don't know if your roadmap was always to build this multi-cloud control plane, but your initial project was working on Rook, this storage system for Kubernetes. Maybe give a little bit of background to why you started working on Rook and what Rook does and how that led you to what you're doing now with multi-cloud.

[00:27:53] BT: Yeah. When we started the Rook project, I think this is November of 2016 is when we open sourced it. What we saw was there was a need for reliable storage on Kubernetes especially in scenarios where you're not in a public cloud. Even there the story was a little weaker back then. So we started – Instead, we started the Rook project to essentially bring complex storage systems and run them directly on Kubernetes so that they could run in a more streamlined fashion and they could auto-scale and deal with a lot of the scenarios that you expect out of a Kubernetes application or a Kubernetes service.

So a common scenario for Rook is you want to run an object store. So you basically, through the Kubernetes API, you could – And through Rook you could say, "I need an object store and here's the configuration for it," and then suddenly you have something like S3 running on your Kubernetes cluster.

Now if you're in Amazon, you're likely using their managed service for object storage, S3. If you're on-premise, there isn't one. So you might as well run something on Kubernetes versus, say, managing storage systems in the all traditional way on physical machines or directly in the OS layer. That was the motivation for what – We learned a lot doing that project. We learned a lot about Kubernetes. We learned a lot about declarative management. We learned a lot about control planes. When we started Upbound, the goal was to kind of extend that mission and vision of being more – Building of a more open cloud, and that led to the project that we announced this morning around Crossplane.
[00:29:48] JM: Okay. Let's talk a little bit about federation. Working with multiple Kubernetes clusters, and then we will get to what you're doing with Crossplane, but I just want to talk a little bit more generally about Kubernetes first.

So I've talked to some different enterprises that like they're going with Kubernetes, but they might have thousands of engineers and there are different teams that are working with different Kubernetes clusters in the same organization. So I think this is probably true at most companies that are adapting Kubernetes. They have multiple clusters. What are the pros and cons of multiple Kubernetes clusters? For people who are listening who they want to familiarize themselves with Kubernetes but they haven't actually worked with it. They might just think, "Okay. This is this big distributed system that managed all your stuff," but actually you might have multiple Kubernetes instances. Can you tell me more about that?

[00:30:47] BT: Yeah. I mean, a lot of people are managing more than one Kubernetes cluster. There are inherent limits to a single cluster or people want to use different clusters for different purposes. Maybe have a production cluster and a staging cluster, or they scale the clusters differently. They might use one cluster for stateful workloads and another one for stateless workloads. There are numerous reasons that why people kind of put a limit on what runs within a cluster. Multi-tenancy is another one.

In this modern age now, it's really easy to create Kubernetes clusters, especially if you're using the cloud provider's managed services. So you could line up a GKE cluster in a few minutes and you have your own Kubernetes cluster. So essentially there are more and more reasons now to treat Kubernetes clusters like cattle. Just as in the past we used to use expression treat your nodes like cattle.

[00:31:47] JM: Cattle, not pets. More dispensable.

[00:31:50] BT: That’s right. More dispensable, more tangible. So we’re getting to the point where you could maybe create a cluster, use it for a short amount of time and dispose of it. Whether you have a few clusters or you have hundreds of clusters, one of the problems with managing those clusters is how do we get consistent configuration across them? How do have a common surface area for management across them? Also how to orchestrate work across them?

This whole multi-cluster effort is interesting and is one that the working federation, Kubernetes Federation, has started around both addressing the administration issue of multiple clusters, but also starting to think about how to deploy and manage containers across clusters, being smart about region, locality, availability and other things that they consider.

[00:32:44] JM: That term federation, what is a federated Kubernetes cluster?

[00:32:50] BT: it’s essentially when you have multiple layers of clusters. So you have a higher layer that is connected to a set of clusters on the lower layer, and the cluster at the higher layer is able to control the ones at the lower layer. It’s a hierarchy of clusters where you have one cluster connected to – That access the federation control plane connected to multiple clusters that are just standard Kubernetes clusters.

[00:33:19] JM: Why would you want to federate a Kubernetes cluster, or I guess a hierarchy of them?

[00:33:23] BT: Because you get essentially a single control plane that’s able to control multiple. So you could do things like, “I’d like to deploy a workload or an application that I want to run across these clusters.” Maybe run cross-multiple regions. So if you’re doing that, you either have to do that essentially one-off, go to every cluster and do that, and if things change, you have to go one-off, change it all, or you use a federation approach where you have a control plane that is orchestrating work across other clusters. Just like a standard Kubernetes cluster orchestrates work across machines and nodes.

[00:34:04] JM: When you talk to engineers who are using Kubernetes at a place like Google or Netflix, are they doing federated clusters?

[00:34:14] BT: There are some folks that are using federated clusters, and I think CERN is the one that comes to mind. I think the last KubeCon they talked about using federation, and they had 250 clusters within CERN. I'm sure there are other cases where people use federation. The project itself has gone through a rewrite and I think there is definitely more traction on the second version of federation now.

[00:34:41] JM: With federation, what is hard about doing that, about setting up – Like if I'm at Netflix, I can imagine if I was able to snap my fingers. I'm like the architect at Netflix. I'm the chief architect. I could snap my fingers and make architectural changes. That does sound pretty appealing having a single control plane basically the internal Netflix cloud. They probably have something like that, some kind of control plane internally, but doing it through a federated Kubernetes cluster sounds like it would make sense. What's hard about setting that up?

[00:35:15] BT: I mean, I think some of the things that are hard about that is that just if you're using federation, Kubernetes Federation, it's still a new project and there are lots of missing gaps and features. But there's also a lot of bootstrapping that you have to do when you get clusters working. I'd say it's not necessarily obstacles at this point. It's just more work that has to happen to get to the level of this thing working in a streamline fashion that say Kubernetes does today at a single cluster level.

[00:35:48] JM: So the ideal scenario might be that I have my Kubernetes top level control plane running in whatever cloud, or on-premises, or whatever, and then I've also got Kubernetes clusters at Google. I've got Kubernetes cluster at Amazon. I've got one at Microsoft, and the one that I have on-prem or the one that I have in whichever cloud provider I want, this federation point can communicate out to the Kubernetes instances, or the Kubernetes clusters at all those cloud providers. Then I've got a multi-cloud scenario.

[00:36:28] BT: That's multi-cluster. Yes.

[00:36:30] JM: Multi-cluster.

[00:36:31] BT: Which could also be multi-cloud.

[00:36:33] JM: Right. Okay. So that brings us to Upbound. We've explored some of these different challenges of Kubernetes and multi-cloud. We talked a little bit about federation. What are you trying to do with Upbound?

[00:36:46] BT: We started Upbound essentially with the belief that there needs to be a more open cloud computing platform. Our mission is to essentially help make a more open cloud computing platform available to organizations and to end users and to the general community around cloud computing. So that's our mission and we're doing a lot of work in the open source community. We think of Rook as the first project that we started in that space. Today we announced Crossplane, which is our second project in that space, and we think as essentially a central piece, if you will, for a more open cloud. Going forward, where later sort of next year we'll be announcing a bunch of other projects that are also within the same vision of a more open cloud computing platform.

[00:37:35] JM: So Crossplane, this control plane for multiple clouds. What's the architecture of Crossplane?

[00:37:43] BT: So Crossplane is essentially a control plane that you could connect to multiple clouds. So multiple cloud providers, or multiple regions within the cloud provider, or even across hyper boundaries on-premises or across clouds. Once you connect your environments and your clouds to Crossplane, you are essentially – You now have a single API and a control plane that has a set of controllers that can orchestrate and manage work across these cloud providers. You're essentially – Think of it as similar to how we talked about federation, but the scope is not just Kubernetes clusters and containers. The scope is most of the managed services that are available to you in cloud providers as well as things like Kubernetes clusters and containers and others.

[00:38:37] JM: So if I'm using Crossplane as my experience that I have a CLI that can do like what we're talking about earlier, you can spin up a Kinesis connection or a Kinesis queue. I don't even know what to call it. Kinesis channel, or interact with Google Cloud bucket storage? Is it a CLI for all those things?

[00:39:01] BT: Yes, and I'll talk about what we use to implement it without confusing the story, but that's right. Think of it as just like every cloud provider has a control plane that offers an API and in front of the API you could use either CLI or their management consoles or tools or libraries, all of those things sit in front of the API. Crossplane is an API and a control plane that is essentially an overlay on top of other cloud providers.

One way to think about it is it's a cloud of clouds, or a control plane on top of other cloud providers. But it offers an API that lets you, like you said, control provision, manage a set of services that could span cloud providers. They're not limited to just one cloud provider, and it offers a layer of portability where possible across cloud providers. So in Crossplane, if you wanted a MySQL database, you ask for a MySQL database. You don't ask for the given cloud provider's managed service for MySQL. So in AWS, it will be RDS, or in Google it's CloudSQL, or in Azure it's AzureSQL. Instead, a developer writes their application and essentially defines their workload, defines their resources that their workload consumes including things like MySQL or message queues or other things without being tied to the cloud provider.

The Crossplane along with admin policy is able to provision the infrastructure based on context, based on policies that are set by the admin and use the given cloud provider's managed services to bind the two.

[SPONSOR MESSAGE]

[00:40:58] JM: Your audience is most likely global. Your customers are everywhere. They're in different countries speaking different languages. For your product or service to reach these new markets, you'll need a reliable solution to localize your digital content quickly. Transifex is a SaaS based localization and translation platform that easily integrates with your Agile development process.

Your software, your websites, your games, apps, video subtitles and more can all be translated with Transifex. You can use Transifex with in-house translation teams, language service providers. You can even crowd source your translations. If you're a developer who is ready to reach a global audience, check out Transifex. You can visit transifex.com/sedaily and sign up for a free 15-day trial.

With Transifex, source content and translations are automatically synced to a global content repository that's accessible at any time. Translators work on live content within the development cycle, eliminating the need for freezes or batched translations. Whether you are translating a website, a game, a mobile app or even video subtitles, Transifex gives developers the powerful tools needed to manage the software localization process.

Sign up for a free 15-day trial and support Software Engineering Daily by going to transifex.com/sedaily. That's transifex.com/sedaily.

[INTERVIEW CONTINUED]

[00:42:47] JM: Okay. So if I'm building my application on AWS today, even if I'm in Kubernetes and I'm thinking, "Oh, I've got portability now. I'm not locked in now." If I use DynamoDB, DynamoDB I believe has proprietary APIs. There's no open source. Like I can't just take that code and just have it run in an open source product, because it's going to run better against DynamoDB, and DynamoDB has [inaudible 00:43:19] APIs. But if you're using Crossplane and you say, "Hey, Crossplane. I want –" You say, "I want a key value store on this cloud over here." The APIs will be more open and they – What? They just translate it into Dynamo APIs?

[00:43:35] BT: No. No. So that's a very good point. There's no silver bullet here. If you're using something like – If your application has built into it a client library for DynamoDB, then you must use DynamoDB's managed service. You can provision it through Crossplane, but on the wire, the data plane, the data path to the actual service is still DynamoDB. So there if you are using proprietary databases and software from a given cloud provider, then you don't get to achieve a high-level portability. You can still use Crossplane to get a universal API and a common way to kind of manage and provision infrastructure across different cloud providers, but in this example, you're still using a given cloud provider's proprietary stack, proprietary software.

[00:44:30] JM: So you're using if I'm using MySQL, for example, since it's open source.

[00:44:35] BT: Right.

[00:44:36] JM: Yeah. Contrast that example.

[00:44:38] BT: It's either open source or has a widely adapted wire protocol. You will see, for example, that people have normalized on using MySQL or PostgreS as a wire protocol for many implementations of databases. So your choices there are MySQL, PostgreS, Aurora, Vitess, even CockRoachDB has a PostgreS – Supports the PostgreS wire protocol. So you get a – There are different ways of supporting and using databases and still not having to change your application when it's consuming different implementations of it.

With the trajectory that we're on, more and more open source software is becoming the default. So we think that we're moving towards a world where there will be more managed services based open software or managed services that support open or widely adapted wire protocols.

[00:45:33] JM: Okay. So the advantage there is let's say I'm getting started with my application and I write – I'm using a MySQL backing for my database and then overtime I'm like, "Oh, actually my application has such a high-volume of data that I want to use Vitess." It's totally fine because you've still got the right APIs. You've still got the same APIs, or if you would say, "Oh, I want global consistency and super resiliency. I want CockRoachDB." You've got the same protocol and this is contrast to what? On AWS today, if I was just going with AWS and I was using Aurora, are there specific APIs, like specific MySQL, like non-MySQL APIs?

[00:46:21] BT: No. So you can run an application that just uses a MySQL client library and it would run against Aurora or it would run against MySQL or run against Vitess unchanged. So that's partly the property that we like or we thought about when we're designing Crossplane, that the decision to use Aurora or the decision to use Vitess or the decision to use MySQL should be a laid-bound runtime level decision, not dev time engineering decision. So if you build your application – And this is not new. In fact, if you look at – We were talking about Kubernetes earlier. If you look at Kubernetes, when I build my application and put it in containers, I don't think about where it's going to run. It's a similar thing, or if it's using a storage volume, I don't really think about whether it's going to come from EBS or from Google Persistent Disk. We think that there is a similar property that could be done for a wider range of resources and services that people are running in cloud today, and that trend is increasing. We're seeing more and more cloud providers offer managed services of open source software and we're seeing more and more commercial open source vendors offering cloud offering, having cloud offerings of open source software, and that's becoming the default.

So if we move to that world, you could argue that what is missing from that is a common way to provision and create this workload portability abstractions so that we could essentially create a separation of concern between what developers do and need and require and what administrators decide to use or SREs decide to use when an application is deployed. That's part of what Crossplane does. It creates this separation of concern between the two, but enables both of them to come together at runtime or at scheduling time.

[00:48:23] JM: Okay. So tell me about that prototypical user of Crossplane. When are they setting it up? What motivates them? Once they set up Crossplane, what can they do?

[00:48:34] BT: Crossplane is still early. We just release 0.1 this morning, but the way to think about it is that it's essentially a control plane. You have to host it somewhere. Right now you could bring it up on your laptop or you can bring it up anywhere you want in any of the managed clouds. Once you have Crossplane running, you add your cloud provider credentials, and at this point an administrator could set some policy for what they want to happen.

For example, they could say, "If an application requests a MySQL resource, I'd like that to be deployed in Amazon using RDS and I'd like it to use this disk capacity, this version, this security group, this VPC." They could set all of those policies that are specific to the implementation of the MySQL generic resource.

When an application developer wants to run an application that consumes MySQL, they don't deal with any of those details. They just say, "I've got an application," let's say it's WordPress, "it requires MySQL. I need MySQL version 5.7 or higher and I want a capacity of 100 gigabytes to start with," and they run that configuration on the Crossplane, and the Crossplane is able to provision both the workload and the RDS database to implement it and connect the two. Open up firewalls, set up security groups, deal with all the things that are needed to enable to get the workload to run without having to bother the developers with implementation details.

[00:50:11] JM: Is it a CLI that translates into whatever commands would be called on the cloud provider that you're asking for? Can you help me understand what is the request language look like?

[00:50:26] BT: Right. Crossplane is built on the declarative management engine of Kubernetes, but don't confuse that with managing containers directly on nodes. We're using the same engine that Kubernetes uses to create a declarative management model. So it follows a very similar approach and that they are essentially its configuration language, like with YAML, you define a manifest that contains your configuration. You could use any of the tools, popular tools that people have used on Kubernetes, like Helm charts or a number of libraries and tools that people have used in that space to define things like a container workload or other types of workload, and MySQL and others, and you define those and you essentially apply them to the Crossplane, and the Crossplane has a set of controllers and a scheduler that is able to take those declarative definitions from a developer, match them and apply the policy that has been set by the administrator, and then the controllers take it from there. They are able to reach out to the cloud using the public API of the cloud provider and essentially provision infrastructure and manage it and set up connections and open up firewalls and do all the things that are necessary to implement what the developer intended.

[00:51:48] JM: So it's a declarative language. It's much like the Kubernetes declarative language, but it's not declaring Kubernetes resources. It's declaring cloud resources.

[00:52:00] BT: We chose that path because we're obviously big believers of Kubernetes' vision, and the Kubernetes team and community has built a very sensible model with really high-degree of modularization. So instead of going and reinventing machinery, we started with that and got to focus on solving some of the multi-cloud scenarios as supposed to reinventing machinery.

[00:52:27] JM: So when I write my declarative file, can I instantiate like a big query system and a Kinesis thing and have all of these things be declared in Crossplane?

[00:52:44] BT: At some point. Right now we support a very limited number of managed services at our first release, our initial release. Plenty more are coming, and if any of the listeners are interested in contributing to this project, it's open source, it's community-driven. We'd love to get more people involved around this project.

Yes, absolutely. The way to think about this is that we are going to be adding more and more managed services and also looking at different kinds of workloads. Eventually, the goal is that we cover a very large percentage of the surface area of the cloud.

[00:53:18] JM: Is this what people use Terraform for?

[00:53:21] BT: Terraform is an interesting project. It enables essentially a devops engineer or an admin to be able to provision infrastructure using a declarative model. It doesn't offer a workload portability as a value proposition on top of it. So it doesn't have, say, a generic MySQL, or a generic definition of a workload. It's a great tool, but it's also designed to run by humans. So a human has to apply configuration, confirm it and do all that stuff. It's not designed as a, say, control plane that has a bit of an autonomous set of controllers that can make changes between desired state and actual state.

[00:54:01] JM: So the idea that I'm seeing with Crossplane, if I hear you correctly, is you write a declarative file and in the long term it will be able to provision any of these crazy managed services on AWS, or on Google Cloud, or an Azure and it will feel like you're importing NPM packages or something, like it will feel as simple and unified as that, but you'll be provisioning these things that today are a real pain to stitch together.

[00:54:31] BT: Correct. You're provisioning them, and it's also a place where there are a set of smart controllers that can automate some tasks across them, and it's also a place that essentially these smart controllers could optimize the placement of where these services can go. As this project grows, you can imagine more and more smarts going into it. I think your NPM analogy is great.

[00:54:59] JM: So has anybody done anything like this where they've tried to make these unified or at least translating? Because what's annoying about this is you're going to have to write translations between the declarative syntax and every single cloud product that people want to use, or they can write their own, but has anybody done that before?

[00:55:21] BT: Yeah. I mean, unsurprisingly, I'd say Kubernetes has done that for both nodes, some of the networking pieces and [inaudible 00:55:28] volumes, and that's partly why we chose to base this project on the engine of Kubernetes and take a lot of the lessons learned from container orchestration and reapply them to multi-cloud orchestration.

But Terraform is another one you brought up. They have a pretty large library of plugins and providers that have done a similar thing. Yes, there're some of these. I don't think anybody's done this and attacked this problem across clouds and across managed services from a control plane standpoint, maybe with the exception of the federation project that has done something similar, but that only has a scope of Kubernetes clusters.

[00:56:08] JM: Got it. Okay. Well, I think we've covered what you're doing today at Upbound in pretty good detail at this point. What's the roadmap look like? You're at 0.1 with Crossplane today. Where are you going from here and how long do you think it will take?

[00:56:22] BT: Well, it's hard to tell at this point. I mean, we're obviously very excited about this project and we think it's a central piece to a more open cloud. So we're going to be investing in it. We're excited that some folks like GitLab and others have endorsed it and want to build on top of it. We're excited to get more people in the community involved so that it can actually accelerate. As far as Upbound's roadmap, this is one of the projects that we're looking at. We hope to come back and talk to you about a few more next year.

[00:56:51] JM: Okay. Well, that sounds great. Bassam, thanks for coming on the show once again, and I hope to see you at KubeCon.

[00:56:56] BT: Yeah, awesome. Always a pleasure. Thanks.

[END OF INTERVIEW]

[00:57:03] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use, and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plug-ins. Use the value stream map to visualize your end-to-end workflow, and if you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on-the-fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily.

Thank you so much to ThoughtWorks for being a longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]