EPISODE 1331

[INTRODUCTION]

[0:00:00.3] ANNOUNCER: Kubernetes is an open source container orchestration service released by Google in 2014. It has quickly grown into a platform with a huge community of enthusiasts and professionals. Besides becoming the de facto standard for container orchestration, it has fostered an ecosystem of related tools and services with increasing power and sophistication.

Argo, a project developed by the company Applatix and acquired by Intuit, is a set of essential Kubernetes native tools for deploying and running jobs and applications on Kubernetes. All the Argo tools are implemented as controllers and custom resources. Some of these tools are Argo workflows, Argo events, Argo continuous deployment, and Argo rollouts.

In this episode, we talk to Alex Collins, Principal Software Engineer at Intuit, about using Argo to manage Kubernetes applications.

[INTERVIEW]

[00:00:57] JM: Alex, welcome to the show.

[00:00:58] AC: Thank you, Jeffrey. Well, good morning.

[00:01:00] JM: Hey, good morning. Today, we're talking about Argo, and Kubernetes and related subjects. I'd like to start with a brief exploration of two unrelated subjects, which is well, seemingly unrelated subjects; Kubernetes, the container orchestration system, and Airflow, which is a directed acyclic graph manager, typically used for data engineering tasks.

These are typically seen as unrelated tools, but I see them as loosely related, at least, because Kubernetes is the distributed systems operating system. Airflow is the directed acyclic graph manager that manages your computation and your execution. These were not made for one another. Maybe you could just give me your brief exploration of modern distributed systems tools and how they relate to one another.

[00:01:54] AC: Yeah, interesting question. I can't speak too much – I can't really speak in more detail about Argo's take on this, and the interesting changes that we find in that area. Obviously, the Kubernetes architecture is very heavily orientated around really, the core aim is to be able to make it easy to deploy web applications that can run massive scale, across multiple machines and clusters and nodes. It's really oriented, very strongly about typically, long-running tasks that can be terminated at short notice. In fact, the termination period on Kubernetes, I think, you have 30 seconds to clean up after you do it.

Things, like Airflow and Argo workflows, they operate slightly differently. They run jobs, or tasks, or batch processing tasks that typically run to completion, and may take a lot longer than 30 seconds to complete the work that they're doing. That can create a bit of friction on the Kubernetes platform. That's one of the main things that you need to address when you're building these software applications. They'll have to do that.

The other thing that is very key to what we do is around the data management aspects. If you're running a batch processing job, or some stream processing job, and then data is really important to you, and it's not going to be typically a small, ephemeral HTTP requests, you'll often be working on quite large files that need to be moved around, and that's obviously expensive, moving data across a network. You're not going to want to stop halfway through processing one of those files. You're going to want to process the whole files. Yet, you've always got this background risk that your process might be stopped by the system infrastructure, so that it can use that processing capacity for some other more important job. I think, that's one of the main things that is a current challenge for software like Airflow and Argo, and to a degree, Tekton running on Kubernetes.

[00:03:54] JM: Tekton running on Kubernetes. Talk about that in more detail.

[00:03:59] AC: I mean, Argo is not the only piece of software running that does task-base workloads. That's particularly Argo workflows, rather than Argo CD. Tekton also does that, but that focuses much more on a CICD use case. Again, it's another example of where you might have a long running process on Kubernetes that you typically want to run to completion, but

again, has this threat hanging over it that it may be stopped midway through the process, and you need to build in various mitigating actions for that. You can do that in a few different ways.

One way is to have various types of retry on that tasks. If it fails the first time, you try again, or you use some memorization strategy to break that task down, so it doesn't necessarily need to be run again. It can just start again, halfway through. Resume strategy. Or there are things, like a pod disruption budget, something like Kubernetes core primitives that can be used to defend that as well.

[00:04:52] JM: Can you just talk a little bit about how the impetus for Argo came about?

[00:05:00] AC: Yeah. I mean, I can give you a bit of history, probably. Argo, actually originated as a product at a company called Applatix, in around 2016-2017. That company was acquired by Intuit, looking to accelerate their cloud journey. Then, a number of other companies have joined over a period of time as other key contributors, I guess you could call them, including Black Rock, Red Hat and Code Fresh, just to name a couple of the bigger names in the area.

Now, the original product that was marketed was basically, a workflow engine for Kubernetes, because not much existed at that time. That's Argo workflows. Probably, maybe what's slightly better known as Argo CD. Argo CD is a cloud native GitOps software application. Do you know what I mean by GitOps? Should explain that?

[00:05:53] JM: Yeah. I have a lot of issues with that word, but I'd love for you to explain what it is to me.

[00:05:58] AC: Fascinating. Sometimes, I think, I usually quote my colleague. He says, GitOps on Kubernetes is git clone, followed by kub CTL apply, which is, I think it really underplays it, really understates, because it doesn't really talk about why those two things combined together works really well. Unnecessary, the word GitOps is necessary. It's some configuration ops. Git is obviously a popular version control tool. A version control tools bring some really useful out of the box capabilities, such as, things like auditing, so you know who's made a change, you know when that change is made, you know that there was some security process about that, so you know the person who was authorized to make that change.

Then, of course, it gives you a history, because you've got your Git history. That allows you to, if you have two commits in your Git history, one that represents the state of the application you want tomorrow, the one that is today. If you roll out that new version, and it doesn't go particularly well, then you can simply roll back to the previous version. There's no manual process to do it. It's typically a very pretty straightforward click of a button. I don't think that really was necessarily the case yesterday a couple of years ago.

That's really why git brings the value wouldn't necessarily need to be Git. You could have some other system in place to do that. The other half of that puzzle is that you're running on cloud native, on Kubernetes native, if you prefer. That brings all the things that are really useful with Kubernetes, such as the fact that you can store resources, you can build controllers, you've got the Kubernetes R back, and you've got all the scaling characteristics that Kubernetes combined.

When you put those two things together, you just have a lot more than the sum of the parts. Argo CD adds some additional items on top of that, things like multi-tenancy and multi-cloud capabilities. With Kubernetes today, as it is, I think, everybody's happy working with individual clusters. When you look at things like the talks that have been accepted to Kubon in a couple of months' time, you can see there's a lot of multi-cluster stuff in that area. I think, that's going to be really interesting area, just generally in Kubernetes. Argo CD effectively has to support that from day one, because you'll have one management cluster, and you'll have a number of other clusters where it stores that application. What is your concerns with the term GitOps? I'm fascinated?

[00:08:18] JM: I don't really know what it means. I mean, I guess, it means that every time you push to git, it's supposed to kick off an entire workflow that pushes that code to production. Is that the concept?

[00:08:27] AC: Yeah. Yeah, I guess. Yeah. I mean, a redacted form. Yeah.

[00:08:31] JM: I mean, isn't that the same thing as continuous delivery, or continuous integration to more – Continuous integration and continuous delivery, like, whenever somebody

tried to tell me that those were separate things, I got frustrated. GitOps is just a rebranding of the same thing.

[00:08:45] AC: Yeah. You just don't have one without the other. You don't package your application, or the expectation you're going to deploy it into an environment. You can't deploy application you haven't package, so you do have the two. The same time, it is interesting how you have always seem to have this handoff between the two. I wonder if it's because the tooling for CI is often very different to the tooling for CD. In your CI process, you're going to have a compiler, and you're going to have a test framework and a bunch of test infrastructure, and then you're going to push that, the result of that is just a binary typically, that goes into a repository. The CD aspect is just taking the binary back out of the repositories to get onto computers. Yeah. You have you don't have one without the other. Definitely not. Yeah. Maybe it's a bit of an artificial line, maybe.

[00:09:27] JM: What's a typical use case for an Argo Workflow?

[00:09:31] AC: There are two answers to this question. The really general answer is that Argo Workflows is a completely general workflow execution platform from Kubernetes, and doesn't really have anything that's perhaps, different to any other systems on there. Its features is because it leverages that ability to scale, you can build very, very large workflows that fan out, particularly wide into thousands and tens of thousands of tasks within your workflows. That makes it suitable for a number of tasks inside the machine learning space.

It makes it a great component put into an existing machine learning application. A lot of users of Argo Workflows, probably don't know they're using it, because they're using something like, Kubeflow pipelines, which has Argo Workflows embedded in it. There's a number of such and such a flow pieces of software that have Argo workflows within it.

The other thing that Argo Workflows that does particularly well is it deals with large amounts of data particularly well. It's got really good integration, really good first-class integration with various bucket-like storage is on Azure, Google Clouds, and obviously, on AWS. If you're working with very large files in the gigabyte range, it just makes that very easy to do on Kubernetes platform.

A lot of people use it for ML. We also see plenty of things in the infrastructure automation space. Because you're already working in infrastructure, if you're on Kubernetes. You need a tool to automate it, you're going to reach the very first tool that's available to you there, so Argo Workflows makes a good fit for that use case. We also see, and then I'd love to see more of this, people using it for CI. We're stepping back a bit and talking a bit about CI, that people are looking to move off Jenkins onto something a bit more cloud native. Maybe they don't want to go to Jenkins X. People are using that. That's a bit more of an edge use case, I would say.

Then we see a lot of people using it for things like, ETL, batch processing and just general data handling. They typically tie in – I mean, Argo Workflows. We have in the Argo universe, we have four bits of software, and they get broken down into two groups as Argo CD and its little brother, or little sister, Argo Rollouts, and there's Argo Workflows. Argo Workflows has a little brother, or sister called Argo Events. Argo events is around triggering those workflows, or determining what criteria that causes workflows to be triggered.

There's a number of different ways you can do that. Quite commonly, we mentioned, obviously, Git commit, but also, web hooks and so forth. That allows you to easily set up a CI process, if you want to do that with Argo Events and Argo Workflows.

[00:12:08] JM: What you're describing here is a fully composable workflow management system for event-based programming, effectively. How does Argo help a typical developer who's working the Kubernetes ecosystem, what problem does it solve for me?

[00:12:29] AC: I think, when we speak to our users, they really come back with three or four things that are really important to them. The thing that underlies this, I think, is the fact that people want to use – when they're using a piece of software, they want something that's typically quite fast, and they want it to be reliable. Those are often the most important things to them.

The speed with Argo Workflows comes from the fact that you can do these big fan out workflows, and you can offload all that work on to some cloud infrastructure that can scale up to the work that you want to do it. The reliability comes from the various features that are there

inside the product, that allow you to deal with that ephemeral nature of running on Kubernetes. Those the two things that bring it together. They make actually quite a good developer experience. This is what we hear from developers.

We asked them, why didn't you use this piece of software or that better software. The thing that often comes back really strongly is, it's very easy to get started. They just had a great experience first time using the software out of the gates. They were quickly able to get what they want done, done. They didn't have to necessarily learn a lot of new things to do that, because they're already familiar with the Kubernetes platform. All that concepts and ideas and things that they already know are already there. They know how to deal with the R back. They know what a deployment is. They know what a pod is. They know how to get their logs. They know which logging facility their logs are in.

All those things, they don't have to learn in these at all to use Argo Workflows, or in fact, Argo CD as well. That just makes it really easy. I don't know if you've – Jeffrey, if you've ever tried to install a software application, just to try it out to see if it will solve a problem. You're there two days later, just getting the very basic setup done, and you want to just run the hello world example of it. That's an amazingly frustrating experience.

It makes you feel it reflects badly on your ability as a software engineer not to get this popular bit of software working. It also is a very frustrating experience, it's going to push you away from using that bit of software if it doesn't work particularly well, first time. I just think that's – it's really easy to understate the importance of that. I think, it's really easy to understate that. I think, people often focus on features and functionality. Actually, that developer experience, that user experience is incredibly important. I think, that's the one thing that we really bring to the table that sets us apart from other similar projects.

[00:14:54] JM: You're at Intuit. Intuit has a lot of complex workflows. Can you tell me about how Intuit uses Argo and Kubernetes?

[00:15:02] AC: Yeah, I can. Intuit is on many companies, a journey to cloud native, from previously running a lot of the software in-house I can't talk too much in detail about propriety systems. Going into the cloud and using cloud resources, such as databases that are available,

to just get things done more quickly. Intuit has built its own internal Kubernetes system called IKSM, which stands for Internet Kubernetes Service Manager, along with a developer portal. Intuit is not going to be alone in having done this.

With the goal of just making it almost a click of button process to get a skeleton application that you can – that has all the services necessary with it, and you can say, “Well, I want to connect this application to an Event Bus. I want to use some identity-based services, and I want to use these security things. Any of these kinds of data storages, and a shopping cart where you select those and you click a button.” You give your service a name, and 20 minutes later, that service is up and running.

It's got all the CI and CD pipelines set up for it. It has skeleton sets of unit tests and integration tests available. It’s in whatever language you're particularly interested in. Obviously, Java is a popular choice, but we also do GoLang as well. It's just all there ready for you to do. We also build on top of that, to build a batch processing platform, so that's something that's being built at the moment, and a machine learning platform, that again, these both leverage technologies, like Kubeflow and Airflow to build those different platforms, as well as some other proprietary technologies to do it. That's all migrating onto a cloud native platform.

Things like operators are pretty commonplace in teams building their own operators. One example I can think of is an operator who's responsible for making sure that the correct IAM roles are available for your application, and you need to have the right IAM roles to be able to access the data that you want to have, so there's things that deal with that as well.

Argo Workflows and Argo CD set really key as part of that vision. Argo Workflows is we install that on every single one of the clusters that we run, and we run several hundred Kubernetes clusters. Users can just, as part of the application, if they want to run a workflow, they can just run the workflow. There's nothing there for them to do. They don't need to install Argo Workflows and stuff. It's a managed solution, and they get regular software updates and upgrades all the time.

Obviously, with Argo CD, they have the tooling available to deploy that application already. They just need to go in and click a button. When they want to do an upgrade, again, it’s a click of

another button. A lot of that not necessarily make work, but not particularly interesting work, where you're not bringing the value to the customer that you want to be doing. You want to be solving that end-user, the end-customer's problem. You don't necessarily want to be worrying too much about manifests and cloud formation templates and those kinds of things. You want to be writing actual code that brings actual customer value. The whole thing is really geared and oriented around that.

On top of that, of course, Kubernetes as said, is a great platform for dealing with processing data. Running Web Services is really great at that. There are additional services built on that, around things, like data lakes and event buses, that allow you to then integrate with those as well, if that's where the data that you need to operate is there. I think for every organization, it's an ongoing journey as well. It's not something that just happens overnight.

It's going to take you at least six months, or years to get through that process and get to where you want to go. Maybe when we get there, that's all changed, and we need to now go off in a different direction to do something differently.

It's pretty exciting to be involved in this, to know that you're working on some of the – you're working on tools and technologies that are used by hundreds upon hundreds of developers, internally in your organization, thousands upon thousands of developers in the wider open source community, as well. Actually, it's really interesting, really exciting to work on.

[00:18:54] JM: You've been in the software industry for a while. How do you think the Kubernetes ecosystem has changed the development of software?

[00:19:01] AC: I think, that there's a lot of really clear delineation with Kubernetes that makes life easier. I have been in the software industry for a long time. Thank you for reminding me. Makes me feel like an old man, and I have had a conversation about punch cards early in my career. That did happen. When you originally deployed software, often it was a case of just copying a binary onto another server, every software engineer in the team having access to production service. It's a risky process. Nobody really wants to do that. We want to be able to have gates and approval processes in place for our own security.

We want to know what happened when and we want to know why it happens. That's probably the biggest changes, and we've gone from this building the binary on your desktop machine, using SCP, you copy that to another machine, and then deleting the existing one with RM and copying the new one and performing a service restart. That is such a manual process, and such a risky processes, and such an error-prone process.

When we originally got Docker and the ability to package any software application, you can largely language agnostic into a binary, that then run on any Linux, Unix host, that was fantastic. Suddenly, you're in a position to make it very easy to do. Early in that Kubernetes journey, people were – I've been using Kubernetes since version 1, 3, I think. We're on 2, 120. 15% through the Kubernetes journey.

People didn't understand some of the things were happening in a software application. I can again, recall telling you a war story here, Jeffrey, I can recall having to Google the term 'crash loop back off,' and I found no results for the term crash loop back off. I didn't know what it meant, and I couldn't figure out what that meant. We had to go and figure – the application was crashing, and it was in a loop, and it was backing off trying to restart it, because it was in a loop. Now, every developer knows exactly what a crash loop back off is, and obviously is terrified of it.

Kubernetes makes that really easy for people to do, provides the R back that I mentioned earlier. It provides a declarative way to say, what do I want my application to look like? Then provides really well – this is something, an approach I mentioned earlier. A really good documentation that's really easy to understand and really easy to read, that is incredibly valuable. I don't want lots of documentation. I want the minimal amount of the right kind of documentation to understand it.

Kubernetes just puts all that documentation out into the public domain. Any developer can use, and any developer coming into a company that's using Kubernetes can just get started really quickly, and they can avoid doing that. I think, that's one of the great things about where we are today. I'm really fascinated by technologies, like Autopilot from Google. Do you know Autopilot?

[00:21:53] JM: No. What is that?

[00:21:55] AC: It's a new version. It's a pretty new version, maybe a month or two old of Google Cloud's Kubernetes service, that basically reduces the amount of operational work that you have to do. That's what's great about it. That's the benefit, I don't have to worry about operations. It scales everything up and down for me automatically. I don't have to install the metrics, and so forth. I think, we're going to move more in that direction of a more managed solution on Kubernetes, where users don't have to think too much about things, like right-sizing their applications, getting resources right, structured right.

I think, this is all just going to become meat and potatoes, tofu and potatoes for developers, and they can go and focus on those things. Because, again, ultimately, my goal is get it done quickly, get it done reliably. I don't want to spend too much time and effort on that. I think, that's how we're going to see it evolve over time. I think, it'll become a stable platform as well. I could see fewer Kubernetes releases over time, because when application is mature and stable, it doesn't need too many new features. You don't actually often need new software releases of it.

[00:23:00] JM: Can you contrast the post-Argo Kubernetes world with the pre-Argo Kubernetes world? What did I have to do before Argo that was super painful?

[00:23:11] AC: I mean, just when it comes to Argo, see that product didn't really exist. Argo CD came along at the same time as Flux. The pre-Argo CD world is the pre-Flux world. There wasn't really that tooling for doing GitOps on Kubernetes. That didn't really exist. You'd have had to do that all manually. I mean, maybe you would have been sensible enough to manage your configurations in Git. I think, Argo has been straightforward for many people to do. You would had to build that software tool, or that software agent, who would have been responsible of getting your Kubernetes manifests out of Git, checking that code out for you and applying it to cluster. Then, adding all the R back and reporting aspects to that, that you needed all the multi-cluster stuff.

With Workflows, I mean, there was – You could have used things, I guess, like Mesos beforehand, but they're not cloud native. They don't have that first-class integration. One of the nice things about Workflows is that I can check the status of my workflow from an API and anybody else, also, using those same Kubernetes APIs can get that same information as well. That just makes that integration interrupt pretty easy.

Beforehand, you had things like, Kubernetes jobs. A job is much more simplistic and restrictive and less flexible. It wouldn't have made it easier to do those massive fan out workloads that you can do. You could have done them. For example, by running a deployment with a large number of replicas, you could have done that. Then each replica, understood what replica is and what work it was doing. You need to then build out all that logic to do. That's part of the value that we bring to it, which I think is hidden in your question.

The other thing that particularly Workflows brings to the table that is not really obvious, is that it deals with – and some features to Kubernetes that don't really exist in Kubernetes. This is the same with Tekton, in fact. It has the ability to stop a specific container, or terminate a specific container within a pod, without actually deleting the pods. That's not something that's actually possible in Kubernetes today. Capabilities such as signaling processes within a pod, again, it's not particularly – it's not available as API within Kubernetes today. Managing things that are pod level, which is you need to do when you're running tasks, rather than running software applications is one of the things that it brings as well.

[00:25:21] JM: Tell me about the engineering behind Argo. What has needed to be built?

[00:25:26] AC: All the Argo suite of applications are built on a pretty standard technology stack. On the controller back-end side, we use GoLang and GRPC. On the front-end, we use React and things like a SCSS, and so forth. On top of that, on top of those layers, we've built out on the front-end side, a whole library and suite of components around rendering software applications in the browser in real-time.

Both an Argo Workflows and Argo CD, if you go into the user interface, you don't see the application as it was yesterday. You actually see it as it progresses, as it changes over time as bits and bobs are added and removed from the application, even if you do it, did add and remove them yourself. That uses push-based browser technology to do that, and then animation in there as well to help the user see that.

I think, there's a lot of bench strength to build a good user interface. You need to build an experience that people understand. It's not just about just presenting the application, or the

software resources in a graph. It's about what information is really important to the user at the time that they're using that particular page, what kind of operations and actions do the user want to take when they're looking at that. That's the top layer in the user interface.

Then beneath that is an API for it. The user interface uses the API. It's an API-first design, which I think is probably the default for people now, and probably goes without mentioning in most cases. That provides a set of rest-ish, rest-like endpoints over HTTP that you can use. Or alternatively, you can integrate them with using GRPC, if you're running HTTP 2. That's the next layer down of the application.

Then below that, we use GoLang. That's grounded. GoLang is a relatively new-ish language. I don't know, maybe new-ish 10-years-old, I guess that's new, isn't it? That contains a whole bunch of libraries and bits of code. We really lean really heavily on a particular Kubernetes concept called an informer. The job of the informer is basically keeping an in-memory representation of what's going on in the cluster. Then allow you to integrate with the informer as a software component, and listen to changes in your cluster and perform operations on that. That's all stuff written by the Kubernetes team. That forms that that layer, obviously. Then below there is Kubernetes and Kubernetes R back that we utilize and use there.

That's really common. I mean, we talk a little bit from a technical point of view, the key components in our software applications are a sophisticated Kubernetes operators. An operator is basically, a bit of software that sits in a wait loop, listening to changes in the cluster and then reacting to them, which is a simplification. There's a whole load of other things going on there as well, as whole load of different bits of logic dealing with various scenarios, such as pod being deleted unexpectedly, or there being an issue with connectivity, all the normal things you need to deal with a software application, they're all rolled in there.

We actually have a library. We have to provide a lot of that logic that we share across all the software applications. We just call it PKG. It's a shared library that gets really reused in several different places. I talked a bit about user interfaces, necessarily use cases, put in buttons and text boxes on a page. It's also understanding how the user interacts with that. We have lots of processes where we sit down with our users, and we get them to show us how they use the software, just in their day-to-day, and we look at that, how they operate with it. We look at the

same way that they do that when they use command line tools, automation tools to understand how they use those tools, so we can make sure that we build out the right tooling underneath it.

These same bits of important knowledge apply to the back-end. The back-end code, I guess, it's not just the lines of source code. It's also the knowledge and understanding of software engineers have of that code. Sometimes that's explicit in comments. Sometimes, it's tacit, tribal knowledge in their heads. I don't think you'd ever avoid that tribal knowledge issue. You can only reduce it. There's plenty of information in people's heads that's really incredibly important to understanding how that operates under certain circumstances.

That means that a software engineer can often, quickly answer a question about why is it behaving in that way? The answer may be, that's how it's designed to behave. That almost forms part of the software application in my mind.

[00:29:45] JM: How do I define an Argo Workflow?

[00:29:48] AC: What you need to think about, the work that you want doing. You need to think a bit about how that gets the data from data storage, depending on what that is. Then the workflow itself is defined as a Kubernetes custom resource. A custom resource is something like a deployment, or a replica set, or a pod, except it's custom. Its specification is defined by the organization that has designed it.

Our custom resource for Argo Workflows is called a workflow. That basic has some metadata name, the namespace it lives in when it was created. Then, it has a specification, which is in simplified, it's a mixture of some configuration about how it should work, when it should be garbage collected, plus information about the work it should actually do and how that work should be ordered. That's typically specified as a series of tasks. Each task is a template underneath. Templates, I mean, often, things just boil down to positing containers on Kubernetes, and that's exactly what Workflow ultimately boils down to, is it's – here is the workflow specification, but it boils down to run these pods in this order. Wait when this pod has finished, start that pod. That's the real nuts and bolts of it.

[00:31:03] JM: What are the parts of the application of the Argo system that need the most improvement right now?

[00:31:10] AC: On the Argo CD side, I think it is managing big applications, so applications with hundreds of different resources. Also, it's the thing that hasn't worked particularly well in the past is the thing called an app of apps, which is a term that's come from a community. It's basically an Argo CD application that's made up of other Argo CD applications. It acts as Uber application. That's often used when people want to deploy lots of very similar versions of the same application with slight configuration tweaks.

This is that many, many clusters at scale problem that some users need to deal with. They need to have many, many versions of the application. They're all slightly different. They've got some slight different configuration, maybe around some TLS certificates is quite a common example. Maybe they have a pre-production version, which has lower resource requirements in the production version. They need to manage a hundred of them, and they've got a configuration management scale issue there. They've got so many different manifesting YAMLs need to do that.

That's being addressed by a new feature that is – I think it's still in alpha, or might be in beta called an application set. An application set is we can see how it's derived from the app of apps idea. It's a set of applications that are all brought pretty similar now. They typically live in different namespaces on different clusters.

On the workflow side, our focus has really been on reliability and performance. The performance aspect is we found that users are now running at a much larger scale. Really very large companies are now advanced in their Kubernetes journey. Companies that don't even talk about it. You wouldn't even suspect, unless you knew people working in these companies. They were doing that stuff. Because of the size of these companies, everything's done at a really large scale. They need to run large numbers of workflows at scale.

We've had to do a lot of work to make that performant and a pleasurable experience, including writing lots of documentation, best practice and then implementing new features for that. I think, we're actually really good state and the workflows area. I'm not actually expecting too many

changes in that area, apart from improved integration with other services, which is computationally quite expensive to do. I won't go into the boring details. I'm hoping that we're going to be bringing a new product out, and that works quite well with Argo Workflows, called Argo Dataflow.

Argo Workflows is orientated, really, about running single workflows that will finish processing some data, and they're done. Whereas, Argo Dataflow is orientated around a workflow that might run forever, and has a different architecture underneath it as a result of that. I think, we had a lot of people asking for a streaming workflow, or a continuous workflow, or a workflow that can be driven from Kafka. The underlying paradigm of a workflow is not really amenable to that. Doesn't fit particularly well. Whereas, Dataflow is specifically designed to address those kinds of use cases.

Because it's part of the Argo ecosystem, I'm hoping these will work pretty well together. They all use the same APIs and same software. You can have a workflow that triggers a data flow, and an event that triggers a workflow, and all these things interrupt particularly nicely. That means that if one of those tools doesn't solve your problem, straight away, you can then just – as we have another tool in our toolbox for you that solves that problem for you. You don't have to necessarily learn a lot to get started with that. That could be a really good experience for you.

I think, also, multi-cluster. In the machine learning space, people are looking for the ability to train their models in one place, because that particular cluster might have some particular resources, like a lot of GPUs, for example, they make it really good for training a particular model, but they actually want to execute the model in that cluster, because they don't need all that expensive GPU resource.

They want to have, well, you talked a bit about CI and CD. What they want to do is do the CI part of ML in one cluster, and then do the CD bit, getting that model into production, and that goes into another cluster, in production cluster. That's a hurdle now to get between that. With a multi-cluster workflow, which I think is the next big thing on our roadmap for workflows, that will make that really easy to do, because you'll be able to define a workflow that does precisely that, that starts in one cluster, and does that initial process and finishes in another cluster.

I think, that will actually open up those really interesting new use cases. I mean, I can't even predict how people will end up using that, if I'm honest. I'm hoping. I'm hoping I'll be really surprised by some really interesting things that people do with that. That's what I'm hoping.

[00:35:56] JM: Could you talk in more detail about how Argo compares to some of these other workflow management-like systems, such as, what's the one Sequoia had? Temporal. Have you seen that one?

[00:36:07] AC: I don't know that. I know other similar solutions, because we have a lot of A versus B versus C comparisons blog post on the Internet that compare that, because that's a popular blog post. People are really, "I'm interested in that. I want to know how we square up against other competitors, I guess." Maybe they don't see it that way. Things like Prefect and Luigi and Airflow.

Typically, things, just they do fit different use cases. Definitely, Argo Workflows isn't necessarily the right solution for many use cases for certain people. You might want something more sophisticated. It might be the right thing for you at the beginning of your journey and may continue to do the right thing for you, because of its simplicity. Then, you might want to get something more out of the ecosystem. Kubeflow is a great example. Kubeflow is a really sprawling ecosystem of machine learning tools on Kubernetes.

There's one particular tool I mentioned, the Argo Workflows has used, which is the Kubeflow pipeline aspect, but it makes it really easy to ladder up from there. If that's what you want to go and do, you can start out with something simple, and you can go for something more complicated later on. It's just going to be different things for different teams, depending on what they want to do in that area.

On the Argo CD side, I mean, the other bit of soft that's very similar to Argo CD is Flux, but they have different design philosophies, and line them out. I will be cautious talking out of turn, but Flux is a bit more opinionated about it. It's less user experience, user interface focused than Argo CDs. For some people, that may be preferable. Philosophically, they might align better with what they want to do. Argo CD has a number of – has a very big ecosystem. Lots of different companies involved in that. That may be that what's important for you when you're looking at

what goes on in the area. If somebody only tells you what's great about their product, and they don't tell you the downsides, they're not telling you the whole story.

[00:38:03] JM: Right. It sounds like, what you're saying is, the days of workflow orchestration are young. We have many, many more years of workflow orchestration to develop.

[00:38:15] AC: Yes. I think so. Yeah. I mean, it started in the 1960s, and I don't see it going anywhere soon.

[00:38:23] JM: Yeah. This idea has been around for a long time. Basically, the concept that you have several different computers, the computers need to work together to accomplish something, and the Workflow is the idea of accomplishing something across those different computers.

[00:38:36] AC: Yeah. I think, it's almost fascinating that it looks like, you're reinventing the wheel each time you do it. Why didn't we set a lot of technology back in 1971 on this, but new problems and new systems come along, that need to be addressed in different ways. People might have different requirements. It may be enough, in some situations that I just want to run my job on a particular schedule, so a cron job might work perfectly fine for me. I might have some very different requirements in other situations.

[00:39:08] JM: You're going to Kubecon. What are the developments in the community that you expect to see?

[00:39:13] AC: I think, it's going to be really interesting. With COVID times, it really changes how you experience something like Kubecon. Those chit-chats that you have, and those accidental call away conversations is a tautology. There's a much harder, and I think that's going to be a thing that's going to – a lot of people miss from that. I think, we'll learn all about that.

I think, we'll hear more about data, and more database discussions at the conference. I think, we'll hear more about multi-cluster. I actually don't know what people will be talking about in the multi-cluster space. I think, we'll hear more of, again, more about data and machine learning and AI running on Kubernetes. I'm hoping we'll see some really interesting use cases, from

companies using Kubernetes in ways that you don't really know about. I think, we're going to hear a little bit about that.

I'm going to be speaking there. I'll be speaking with Jason Hall. We've got accepted. We'll be talking a bit about lifecycle management of containers on Kubernetes, and about how we do that, and how the lessons that we've learned from building applications on Kubernetes that do things, are a bit esoteric or unusual. Maybe a bit on the edge, and hope that people will come along to that, and hope that will be really popular. I certainly find that particular topic fascinating. I love hearing the sound of my own voice, so that'll be really fun. I wonder, if people be talking about developer communities as well, developer experience. That will be interesting to see.

[00:40:42] JM: Yeah. I mean, I've been watching the Kubernetes ecosystem for, I guess, four years now, and five years really, and it's gone to become this – I mean, it's just a gigantic – I don't even want to talk about the cloud native landscape or whatever, the diagram with bajillion products on it. It really is comprehensive and increasing, covers everything in distributed systems.

As we draw to a close, I'd love to get your perspective for Kubernetes is not – For me, it's hard to know, what are the developments that are going on within Kubernetes itself, as opposed to the developments around it? Meaning, I see lots of developments around it, things like Argo, Istio, Envoy, Calico, right? That's the networking interface thing, EBPF, whatever. There's a bunch of these things. Is there still a lot of development going on in Kubernetes itself? What kinds of advancements are taking place within Kubernetes itself?

[00:41:38] AC: I'm going to say, I would struggle to answer that question. They've recently reduced their release cadence. I'm hoping that it becomes a utility. It doesn't actually evolve a great deal over time, because you're building applications on top of it. You don't necessarily want the API is to be deprecated particularly often. You want to keep them in place to keep your application that uses those APIs. I think, that would probably be my hope for that core system is actually, it will go into a maintenance mode, where we can all trust and rely on the features that are available to it.

I think, anybody who's worked with Kubernetes will tell you that, I mean, they do an absolutely fantastic job of that. It takes a very long time for an API to be deprecated. There's always a very clear reason for it, and a clear path to achieve the goal that you wanted to achieve with that particular API. That would be my hopes for the ecosystem. There are some things that are really – that you spend time on that you don't necessarily want to spend so much time on. Things like, metrics and logging, and reporting on the internal workings of the cluster. How well are you using your resources? How is that utilized? A lot of this stuff isn't really – there's not really much in the core of Kubernetes around that. It's all stuff that's built on top of it.

I feel, we'll probably end up with a situation, the number of technologies that people use in those spaces tends to become a bit more – that ends up being one particular technology that people gravitate towards, and that they use. I think, we're seeing that with Prometheus in the metrics space. Prometheus doesn't get bundled with Kubernetes, but you're probably going to be using it with Kubernetes.

In terms of packaging, your application manifests, tools like, customize and helm are very dominant in that area. Again, customize is quasi-part of the Kubernetes core, but that helm isn't. We don't see too much evolution in those spaces. I guess, that's what I think will happen. Hope will happen, rather, I think. Well, I hope will happen in this area.

[00:43:46] JM: Cool. Well, that's a good place to close off. I look forward to seeing you at Kubecon. Hopefully, it actually occurs. It's such a dystopian time that I really hope it happens.

[00:43:58] AC: Yes. Obviously, I'm joking about getting out now. I do leave for other reasons.

[END]