EPISODE 1207

[INTRODUCTION]

[00:00:00] JM: In the past several years Kubernetes has become the de facto standard for orchestrating containerized stateless applications. Tools such as stateful sets and persistent volumes have helped developers build stateful applications on Kubernetes. But this can quickly become difficult to manage as in application scales. Tasks such as machine learning, distributed AI and big data analytics often require a distributed application to maintain some sort of state across services. Kubedirector is an open source controller that helps streamline the deployment and management of complex, stateful, scale out application clusters on Kubernetes. Kubedirector provides an application agnostic deployment pattern and enables developers to run non-cloud stateful applications on Kubernetes without modifying the code. Kubedirector is part of a larger project called BlueK8s, which aims to bring enterprise-level capabilities for distributed stateful applications to Kubernetes.

Kartik Mathur is an engineer at HPE Developer, an open source initiative within Hewlett-Packard Enterprise. HPE is an enterprise contributor to the Kubedirector open source community. Kartik previously worked as a senior software engineer at BlueData, which created the Kubedirector project before its acquisition by HPE. Kartik joins the show today to talk about why state is important for big data and machine learning applications. How Kubedirector can help manage the complexity of stateful applications and what's next for the BlueK8s project as a whole.

[INTERVIEW]

[00:01:25] JM: Kartik, welcome to the show.

[00:01:28] KM: Hey. Thanks, Jeff.

[00:01:30] JM: We've done many shows on Kubernetes and Kubernetes applications. Can you define the difference between a stateless and a stateful application?

[00:01:39] KM: Right. So let's dig into an example, right? A stateless is your typical like web server, which does not carry and any state in itself, right? Like a good analogy would be stateless would be like a cattle, right? You can replace one with another. While a stateful is more of a pet, right? Having its own character. So it's carrying a state in itself the services or what's running inside.

[00:02:06] JM: And what are the difficulties of managing stateful applications?

[00:02:10] KM: The difficulty is for a stateful application there are interdependencies between the services. If you take an example of Kubernetes in a microservices architecture, if the applications or the services are stateful, they are interdependent. So the pod migration and stuff becomes extremely complicated because they are stateful, they're carrying states. So you cannot just replace if a host goes down. So that's kind of – It really complicates the orchestration.

[00:02:40] JM: What are the potential solutions for that?

[00:02:44] KM: For the stateful. So one of the solution that we are proposing here is using this operator called Kubedirector, right? So an operator is like the Kubernetes way of deploying applications. So in a typical world an application developer would have to write an operator for their application. What we are proposing is like a generic operator called Kubedirector. And with Kubedirector we are trying to take care of these stateful sort of applications using some of the hooks that we provide and some of the metadata, which gives you state of the cluster to handle state for application. And what we are saying is this is like a generic operator. So any applications built on top of Kubedirector can be stateful and it can utilize those hooks. Without Kubedirector, you would have to write your own sort of operator for your application and kind of duplicate the work across application.

[00:03:42] JM: So let's say I'm running an application like a large multiplayer game. What kinds of problems could I experience if my game is not properly architected in a stateful fashion?

[00:03:56] KM: Yeah. So for example, the biggest one would be failures, right? Or if you lose connectivity between services running on different sort of pods. So in that case, it becomes extremely hard for the game to continue to function if your services or pods are going down. If they are stateless, then Kubernetes out of the box just bring up the pod on another host. But in the stateful, if it is backed by some state, it becomes extremely complicated to handle those scenarios.

[00:04:27] JM: Now let's say I was trying to address this problem with Kubedirector. How would the failures be managed in a way that could be recovered from?

[00:04:39] KM: Right. So what Kubedirector does, it gives you, A, some life cycle hooks. And second, it maintains metadata or the entire cluster view inside every pod on Kubernetes and it is constantly reacting to the change. So as your cluster is evolving or things are very dynamically changing, this metadata is constantly updated inside the pods. And then we have some lifecycle hooks which tells you for a given pod what has happened. Some pod has rebooted or somebody is trying to scale out the cluster or somebody's trying to shrink the cluster. So how do you – Using those hooks, then you can reconfigure your service. That's kind of how we're trying to solve it.

[00:05:27] JM: Let's take this from a top-down view and just cover a little bit about what a custom controller actually is before we discuss what Kubedirector is doing. Can you explain what a custom controller does?

[00:05:40] KM: Yeah. So custom controller is a design pattern, which is becoming extremely popular in Kubernetes world. On a very high-level it is just constantly watching at a given resource, right? And as the resource is changing, it gives you then some hooks so that you can react to the change. For an application specific custom controller, all that intelligence is baked as part of the custom controller or the whole thing is called as an operator. With Kubedirector, we have sort of taken care of all the boilerplate things for you, right? This is one custom controller and then you can build applications on top of this and we abstract a lot of the things.

[00:06:27] JM: Tell me a little bit more about the life cycle of what happens with Kubedirector. So if I'm trying to build an application with Kubedirector, how is the custom controller functioning?

[00:06:42] KM: Right. So with Kubedirectory what we are proposing is – So with custom controller there is a thing called custom resource, right? So custom controller basically extends the API extensions of the Kubernetes. Basically it makes your new resource native to Kubernetes and then it watches and then it reacts to the change. And then you also get the APIs to query what's happening with your resource. So with Kubedirector we are proposing a generic kind of a custom resource. So every application will be utilizing or piggybacking on this custom resource. So if you just do a get on this custom resource, you'll get all the applications built using Kubedirector.

[00:07:29] JM: And can you say more about what else I would do with those resources? Okay. So let's say I've got a Kubedirector instance and I have set it up and I've got a custom resource. What am I doing with those custom resources?

[00:07:47] KM: So this custom resource, basically all the expressiveness for a given application is captured in this custom resource. So like what is the hierarchy for your application? We divided something called as roles, right? So in a complex application there are a bunch of roles. And then every role can have one or more kind of pod running within it. So the structure of all your application is kind of in this custom resource way that Kubedirector defines while the application logic is kind of injected by Kubedirector. So when you kind of built an application using this custom resource, you tell it, "What is the orchestration package for it?" And that is injected by Kubedirector when the pod comes up. And that kind of do the actual orchestration or configuration of the services inside.

[00:08:40] JM: Can you generalize to the types of applications that Kubedirector would help solve a problem for?

[00:08:49] KM: So we are trying to solve the problem of sort of a legacy kind of application. But in theory, anything that could be containerized you could convert it into a Kubedirector way. So basically you follow the custom resource or you follow the examples from the catalog how to define your application. What role should it be? What role has what services and what is the cardinality for every role? And then basically you also give the configuration layer to every role, which configures once the pod comes up. So in short, the answer is uh anything that could be containerized you could create a Kubedirector application out of it.And we have – Sorry to cut you. But we have tried this for most of the legacy application that we had in our catalog prior to Kubedirector like big data applications, Hadoop, Spark or database systems. And now we are moving with the machine learning sort of application with Jupyter Notebook or training cluster or deployment clusters.

[00:09:53] JM: And what has been your experience using Kubedirector for those big data projects?

[00:10:00] KM: Initially it was sort of very kind of limiting in terms of what for a stateful application to correctly configure it to even handle day two operations. But with Kubedirector, as things are evolving, we are continuing to add new features and new hooks. And with those we are expanding our catalogue of applications. So I think we have kind of stabilized a lot of these legacy applications with the features that we already have along with the new features that we are constantly adding to support legacy as well as microservices kind of applications.

[00:10:42] JM: You work at HPE. Why did it make sense for Kubedirector to come out of HPE?

[00:10:47] KM: Yeah, that's a great question. At HPE, currently I’m part of a group called Ezmeral, right? So this Exmeral of today, if we go a little bit of history, basically two startups that came together, which HPE acquired. One is BlueData and the second one is MapR. And this platform that I work for, Ezmeral, was part of the BlueData platform. And BlueData was a container orchestration sort of a platform. So basically we are applying the same knowledge that we kind of gained with our experience at BlueData and try to port it on top of Kubernetes on Exmeral container platform. And that's how Kubedirector came into picture so that we could apply whatever our learnings was to orchestrate these containerized application on top of Kubernetes. And that's when we started this project and thought of giving it back to the community and open sourced it.

[00:11:48] JM: So you gave the example of these big data projects. There have been other efforts to run Hadoop or Spark on Kubernetes. How does Kubedirector's functionality compare to just the normal versions of Hadoop or Spark running on Kubernetes?

[00:12:06] KM: Right. So if you want to run Hadoop or Spark you either have to write an operator, like I said before, for whatever application or for Spark or for Hadoop. With Kubedirector we are saying that you don't have to write an operator. You can utilize Kubedirector. Follow you know the custom resource and then define your application, the Hadoop entire eco system, which could be really complex because it has so many interdependent services.

So once you define in the Kubedirector way of expressing an application, you get all the benefits of Kubedirector without writing an operator. In Hadoop, there are so many services. Usually there are different operators for different services, and you can have so many operators to manage, which could be a headache in itself. So what we are saying that you don't need so many operators. You can just utilize Kubedirector as one operator and build applications on top of that. And it also gives you a very kind of consistent model because your every application is basically of a type Kubedirector app. So your client could be pretty lean.

[00:13:22] JM: Yeah, that makes a lot of sense. Now can you talk through the engineering of Kubedirector? What were some of the most difficult engineering problems in building it?

[00:13:33] KM: So when we were starting Kubedirector, the whole operator concept itself was pretty new. The operator SDK, which is the SDK required to build an operator was still in its early phase. So we had to go through that learning. And then a lot of the ideas that we had how to translate that on Kubernetes in an operator way, that was sort of challenging. And we are not there yet, but we are constantly evolving. And as we are getting more and more contribution, I think we are kind of stabilizing things. It's still not 1.0. So we are still I would say in our early phase. But there are a lot of difficult problems that we have solved and there are a lot of difficult things that still needs to be solved.

So like in a dynamic kind of an enterprise environment, you have a lot of these interdependent applications. So we talked about in an application you have different services which are dependent on each other. You could also have like different applications dependent on each other or connected to each other. So we added features in Kubedirector to make that happen so that if application A depends on something that application B is doing, how to tie them together. And then getting more applications in the mix to build sort of a pipeline, and that's what we utilize to build sort of a machine learning pipeline, which we are constantly evolving by adding new and new Kubedirector applications.

[00:15:03] JM: And so just to emphasize, what is the value that you've gotten out of using Kubedirector for that machine learning pipeline?

[00:15:11] KM: One thing is like either you write operators or you bring a Helm chart in that it will become a very heterogeneous sort of environment. With Kubedirector, your applications are all of one type. So it's easier to tie them together and to write lines to that. Also if you have different operators, it becomes a huge overhead in terms of the resources also to have so many operational parts running and also to keep up with these open source operators. So with Kubedirector, there is only one operator, but your applications can kind of asynchronously change and grow and evolve. So I think this really expedites the process of getting more and more things in the mix.

[00:15:59] JM: And what programming language was used to build Kubedirector?

[00:16:04] KM: So it is Go lang. I think most of the operators are built in Go lang. The initial days, Go lang was the only way to write an operator. I think now they have kind of extended that to other languages. But Kubedirector is written in Go lang.

[00:16:18] JM: Gotcha. And Kubedirector is part of a broader open source initiative called BlueK8s. Can you elaborate on this initiative and the goals and the current progress?

[00:16:30] KM: Right. So like I said, we are still in the early phase, and Kubedirector is sort of the first project like a serious attempt to eventually take it to some level or even incubate it in CNCF. So in that umbrella of project, Kubedirector is the first project and we are doing some work on other machine learning specific operators which we might think of open sourcing in future. So it's still a work in progress and it's a little early to share more details on that right now.

[00:17:03] JM: Gotcha. So this is part of an open source push within HPE. What's the rationale behind the open source involvement within HPE?

[00:17:16] KM: Right. So HPE, especially uh the unit that I work for, Ezmeral, we use so many open source projects and we have always thought that HPE being a large organization, it's kind of our responsibility as well to give it back to the community because we have been a huge consumer so there is a strong push from management as well as from engineering teams to do open source contribution along with having our own open source initiative. And Kubedirector, when BlueData became part of HPE, was already there. From HPE we are getting kind of more urgency and energy to take it to maintain it so that HPE can extend its open source footprint.

[00:18:04] JM: What kinds of limitations have you found with Kubedirector? Are there any problems that you'd like to solve that it hasn't been able to solve quite yet?

[00:18:13] KM: Yeah. There are quite a few actually. So with Kubedirector what we are saying is we are trying to hit the sweet spot. We are not saying that just use Kubedirector and don't use any other operator. It would not have everything that a native operator for an application would have. But with Kubedirector, you could kind of simplify a lot of things. So we are trying to hit kind of the sweet spot rather than going to one of the extreme. So I would say like one extreme is a very kind of a limiting Helm chart sort of thing which is very static or writing a whole full-blown operator for an application. With Kubedirector, for a lot of application, you don't need either of the two. So that's kind of the middle ground as a good starting point to build a Kubedirector applications.

Also in terms of learning, right? Once you have built few Kubedirector application to onboard new application, it's not very hard for a team or for an enterprise. In terms of what remains to be done, well, there are plenty, right? I come from BlueData where we had our own container platform and there are a lot of features which we had outside of Kubernetes in our own container platform that are not there along with a lot of these things that we would like to tighten up or fix. Like we're still trying to find a solution for encrypting secrets, applying placement constraints or policies for role. Also in the security is a huge kind of aspect that we are trying to tighten up. So getting some service mesh so that applications can use it. So we are working on all those things right now.

[00:19:52] JM: So it's 2021. Give me your perspective on the Kubernetes ecosystem as a whole. What are the uh problems that remain unsolved in the Kubernetes ecosystem?

[00:20:05] KM: Right. So Kubernetes, I think although it has gained a lot of popularity, it's still a very, very complex thing for the consumer. It's an intimidating task to switch or migrate your legacy application on Kubernetes. And Kubedirector is an effort in that direction to sort of abstract a lot of these complicated things that Kubernetes exposes. So Kubernetes basically gives you a lot of hooks, and with proper understanding you can do a lot. But I think there is still a huge gap for a consumer versus what Kubernetes exposes. And I think that's where the world will move in next maybe a year or two to simplify things, right? There are so many things available and there are so many overlapping things available. Understanding of things I think will take some time.

[00:21:05] JM: Now as far as HPE, you have lots of I guess clients that have been around for a long time before Kubernetes. And how are these legacy clients adopting Kubernetes. How aggressively are they adopting it?

[00:21:22] KM: Yeah. So I think it is becoming kind of a de facto container platform for a lot of the customers. So we are definitely sensing that. And that's when kind of we also embrace Kubernetes, because we already had our own container platform. And there was a long debate internally whether to take or fully stand behind Kubernetes or continue to support our platform. And I think it's now established that most of the customers are either thinking about Kubernetes today or in long term will be going in that direction. So a lot of customers have already started POCs or trying things out internally in their data centers and some customers are even thinking of productionalizing their workload on Kubernetes. So I think we are definitely seeing it becoming a mainstream in the data center container orchestration platform.

[00:22:20] JM: Those clients, are they having trouble migrating to Kubernetes or figuring out how and when to adopt it?

[00:22:28] KM: Yes, definitely. So migration path is extremely, extremely intimidating especially for large kind of customers, some of huge banks that we have. It's non-trivial to whatever they have been doing. There are a lot of legacy applications plus all the work that have been done to port it on Kubernetes. So it's a challenging thing, but I think they have already started planning in that direction. And that's why projects like Kubedirector kind of helps them because it simplifies a lot of things. Also, like for our legacy customers who have used the BlueData platform, the experience that they get when they build the Kubedirector application. Because a lot of the naming conventions and how things are done, we have kind of retained the same. It's really helpful for them to map it for what was happening with the application when it was not running on Kubernetes versus if they build the same application using Kubedirector on Kubernetes. They get similar experience. So that's why we are getting kind of good feedback there.

[00:23:34] JM: Gotcha. And what else have you learned from working with those large legacy enterprises about what they're doing and the struggles that they're having in terms of modernizing?

[00:23:48] KM: So I think there are struggles at every level, right? So when you have to bring something like Kubernetes in the mix, there are so many uh different sort of personas that gets involved. There is your infrastructure hardcore data center sort of team. There are data scientists. There are software engineers. So it's basically re-architecting a lot of things. And without Kubedirector, if you have to convert a legacy application into microservices based architecture, I think that's a huge undertaking. With Kubedirector, what we are saying is your legacy application can be containerized, but you don't have to really make it like exactly like how a microservices system should be. So in a Kubedirector-based cluster pod, we still encourage running multiple services and we give some hooks for those services to be talking to each other.

[00:24:53] JM: So coming back to Kubedirector, how does Kubedirector compare to other custom controller implementations that you've seen in the ecosystem?

[00:25:05] KM: So Kubedirector, like it is one of the custom controller, right? But in the ecosystem, one custom controller is tied to an application specific customer resource. With Kubedirector custom controller, since it is more generic, we cannot add application specific intelligence as part of that. And that's where our challenge is. Like how to make it so that it remains sort of generic while also the application can inject its own sort of intelligence. And to keep providing that hooks is kind of a very challenging piece for us to have it like as a base, as a generic base for all the applications.

[00:25:50] JM: Gotcha. So you've had to keep it very abstract.

[00:25:54] KM: Very abstract, correct. So even the custom resource, right? So many times it's like we feel that if just we add this particular flag it would drastically simplify things for a given application. But does that flag make sense for another type of application? If it doesn't, then we usually don't do it. And then that's where we have to brainstorm more. How to solve those kinds of problems? So that's why with Kubedirector, what we are saying is you cannot replace or full-fledge at all the operators in your data center, but a lot could be replaced.

[00:26:36] JM: Kubedirector has a number of hooks for day two operations. Can you explain what those hooks for day two operations are?

[00:26:45] KM: Right. So in Kubedirector, the application level intelligence is injected inside the pod once the application pod comes up, right? So Kubedirector on top of that also gives you some data hooks, like if the cluster is scaling out, if a new part is being added, right? In a stateless kind of an environment, it doesn't really mean for the existing running services. But in a stateful world, if a new pod is added, there is a possibility that a service running in some other pod needs to be reconfigured or needs to know about that. So in those scenarios, Kubedirector calls these lifecycle hooks that, "Okay, a new pod of this type is being added," and then the application developer can write their orchestration code based on what's happening there. Similarly if you're scaling down, there are features where you can connect multiple applications together. So if there is any change in those application or given application, they can be made aware of those change and then they can reconfigure themselves accordingly. Those are the life cycle hooks. And I think they're really powerful in giving this kind of solving the day two problem.

[00:28:04] JM: And there's also this continuous validation of resources. So my app, once it's been deployed and it's going to get continuously validated by Kubedirector. It's going to be continually watched. And the resource definition is going to be continually enforced. Can you talk more about the continuous validation of resources and what role that plays?

[00:28:29] KM: That's sort of the operator design pattern, right? The reconciler pattern where you give a given spec and the reconciler basically tries to keep you to that spec. So if there is any change, it reacts and try to bring you back to the spec. Along with that there are also validation web hooks that are in place. So while defining your custom resource, you can define that I want this to be of type that or this. So Kubernetes inherently gives a lot of validation out of the box. But on top of that, if you want to add more validation, we have these validation web hooks using which you can add more validation logic for a given application.

[00:29:17] JM: Are there any bugs that you've had to overcome with Kubedirector around like crash loops and problems with not being able to adhere to the recovery state properly?

[00:29:35] KM: If you're looking for some specific examples, I can't think of any. But those kind of things are kind of common outside of Kubedirector in the Kubernetes world itself where your pod goes into crash loopback states. Sometimes if your – Or your cluster is air gapped, you cannot pull those images that your application is asking for, and those kinds of things. Yeah, those are the ones I can think of right now.

[00:30:04] JM: Gotcha. So if we zoom out again and think about the Kubernetes ecosystem as a whole, do you think people are going to be managing Kubernetes clusters in five or ten years, or do you think there'll be some higher level abstraction that they'll be managing things with or relying on a cloud provider to have a managed Kubernetes piece?

[00:30:28] KM: Right. Yeah, I think that's a great point. So it's a huge undertaking to manage a Kubernetes cluster. And I definitely feel that in a large organization there won't be just one, but multiple Kubernetes clusters of different versions. So it will definitely be very hard to manage them individually. So definitely there should be some abstraction or some layer on top of that. And as we are seeing, definitely one of the cloud providers would come up with a service like that. Or what we are doing at HPE with the Ezmeral platform where we deploy the Kubernetes cluster and manage it and you can have as many Kubernetes cluster as you want. So there will be this abstraction or this management control plane. Similar to what we had for containers, there will be one for Kubernetes itself, and the HPE Ezmeral container platform is sort of in that space.

[00:31:30] JM: Are there other projects within HPE that you'd like to see around Kubernetes evolve?

[00:31:39] KM: There are a lot of promising projects. There is a project by SPIFFE and SPIRE I think that's already incubated in CNCF. It's one of the most popular one. I definitely see that picking up more and more. We might even integrate once we take the identity based authentication approach for Kubedirector applications to integrate that project. But that is one project that I definitely see getting more momentum in coming days. Along with that, the other work that is being done in the machine learning space where we are working on writing operators for model management, model monitoring. I think those would also be – Once they are incubated, they will be popular.

[00:32:25] JM: And what are the other open source projects outside of HPE that you're excited about?

[00:32:32] KM: Well, there are a lot. So currently at HPE, I'm part of the team which is building ML op solution. So there are a lot of promising projects in the ML space that are coming that are building solutions on top of Kubernetes. One of the very popular one is obviously Kubflow, and that has an entire ecosystem of projects around it. To me, it is very similar to what Hadoop was. So in that space itself there is Seldon, Argo, there is Airflow. I think all of those would be very popular and would be used extensively as the world is moving more in the ML direction.

[00:33:13] JM: What do you find so exciting about Kubeflow?

[00:33:17] KM: Kubeflow, basically it gives you an end-to-end sort of kind of a solution for a data scientist. Agreed that it is extremely complex, but the problem it is trying to solve itself is pretty complex. And as it is getting more and more projects, it solves the problem right from getting your Jupyter Notebook to run your training algorithms to deploying using Seldon and also it has hooks to do security to do RBACs. I think it's a pretty solid solution and we definitely have it in our ML ops ecosystem.

[00:33:58] JM: Do you see Kubedirector playing a wider role in the Kubernetes ecosystem? You've touched on a few other projects in the ecosystem, but in what way do you see Kubedirector playing a role in the ecosystem as a whole?

[00:34:14] KM: I think Kubedirector shines because of its simplicity. I mentioned Kubflow. So a lot of things that Kubeflow, we attempted to solve similar problem by building Kubedirector applications. And we were successful to some extent. So I definitely feel the simplicity of Kubedirector will be appreciated once people are serious about Kubernetes with complex set of legacy as well as these new microservices-based application. With Kubedirector, you can really streamline your security by having a consistent RBAC control, by having a consistent layer of custom resources and all the infrastructure pieces that you would need for any operator or any application like encryption of your secrets, policy for your scaling. I think if it is handled by one piece, then I think it would be really powerful.

[00:35:15] JM: And how big is the team working on Kubedirector right now?

[00:35:20] KM: So since it's an open source initiative, we do see contributions from different teams. Also, there are some contributions from outside of the team. So there are like at least five, six engineers, which are spending majority of their time either building Kubedirector applications or enhancing Kubedirector infrastructure. But we are kind of pushing for getting more and more contributions internally and also from the open source community.

[00:35:54] JM: How do you see the project evolving in the next year or so?

[00:35:59] KM: I think the project is gaining momentum since we started adding the machine learning-based applications with Jupyter, with MLflow, with Feature Store. I think it would be very popular for people trying to deploy machine learning-based solution on Kubernetes. Especially, to be able to understand everything that Kubeflow provides on day one is extremely intimidating. So Kubedirector I think will be very handy in the machine learning space because of the catalog of applications it already brings plus the new application that you can easily onboard. Along with it abstracts all the infrastructure, Kubernetes infrastructure sort of things for you. So I think that's where Kubedirector would be ending.

[00:36:52] JM: So just to be clear, the value to that machine learning community is that Kubedirector is going to help with the recovery and resilience of stateful machine learning applications.

[00:37:09] KM: Yes, and also to onboard new applications. If you have any legacy application that your machine learning application needs to talk to, I think Kubedirector gives you necessary features and hooks to make that happen and to build a pipeline and to build a very dynamic sort of system, which is constantly evolving. So it could act as a central piece for those applications or those pipelines.

[00:37:37] JM: Do you see any other outstanding problems in the intersection of the machine learning space and the Kubernetes space?

[00:37:46] KM: So in the machine learning and Kubernetes, like I think the whole area around monitoring of your model and management of your model. So people have sort of written frameworks to do it, but to deploy them on Kubernetes is not there yet. So I think that migration would have to happen. So that is the intersection, I think, where on Kubernetes you do have monitoring for your resources and for every other thing. But what's happening to your model? Those kind of monitoring I think will be happening in next few years.

[00:38:28] JM: Fascinating. Do you yourself spend any time building machine learning applications?

[00:38:35] KM: Yes, I do. If I'm not able to contribute, at least I am very active reviewing the PR or going through the issue list that open source community is adding and then brainstorming. It's really exciting for me to brainstorm the problems that we could solve using Kubedirector. So definitely I'm very active.

[00:38:59] JM: Well, Kartik, I'd love to close off by getting your thoughts on the future. Do you have any perspective on how infrastructure is changing or machine learning applications or Kubernetes? Any insights on the future that we haven't covered quite yet?

[00:39:15] KM: I think we touched on a lot of things, and this is an extremely – I think it's a new kind of stream of science that will evolve because everything will have some sort of a machine learning aspect. If you look at last decade, I think it was all about storing and managing your data and data crunching, data processing. I think going forward, how to make use of that will be what the machine learning would be doing for, I think, all aspects of software.

[00:39:51] JM: Okay. Well, Kartik, thanks for coming on the show. It's a real pleasure talking to you.

[00:39:54] KM: Thanks a lot, Jeff.

[END]