EPISODE 992

[INTRODUCTION]

[00:00:00] JM: Distributed systems are required to run most modern enterprise software. Application services need multiple instances for scalability and failover. Large databases are sharded on to multiple nodes. Logging services, streaming frameworks and continuous integration tools all require the orchestration of more than one server.

Deploying a distributed system has historically been difficult because the nodes of the system must be managed by the underlying infrastructure. If I have a distributed system that I want to deploy the complexity of that deployment is going to be different depending on whether I'm running on AWS, or VMware, or my own bare-metal server infrastructure.

Heterogeneous server infrastructure makes it hard to sell distributed applications that get deployed to that infrastructure. A vendor that is selling a distributed database would need to figure out how to make their database work on the infrastructure of any given customer. Kubernetes Hernandez has simplified the process of deploying a distributed application. Kubernetes is a container orchestration system that has steadily grown in popularity to the point where the ecosystem is mature and the software is stable. Now that the software industry has a reliable, portable means of deploying a distributed application, the enterprise software market is becoming easier to enter for the companies that want to sell a distributed application.

Replicated is a company that builds products for software delivery. Replicated allows for the distribution and updating of applications that would have been hard to deploy in the past. Grant Miller and Marc Campbell are the CEO and CTO of Replicated and they the join the show to talk about the modern enterprise software market and the process for delivering software to companies that might otherwise have trouble consuming it.

Full disclosure; Replicated is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

[00:02:03] JM: Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we're being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications and it's definitely hard to get engineers excited about maintaining those back-office applications. Companies like a Doordash, and Brex, and Amazon use Retool to build custom internal tools faster.

The idea is that internal tools mostly look the same. They're made out of tables, and dropdowns, and buttons, and text inputs. Retool gives you a drag-and-drop interface so engineers can build these internal UIs in hours, not days, and they can spend more time building features that customers will see. Retool connects to any database and API. For example, if you are pulling data from Postgres, you just write a SQL query. You drag a table on to the canvas.

If you want to try out Retool, you can go to retool.com/sedaily. That's R-E-T-O-O-L.com/sedaily, and you can even host Retool on-premise if you want to keep it ultra-secure. I've heard a lot of good things about Retool from engineers who I respect. So check it out at retool.com/sedaily.

[INTERVIEW]

[00:03:40] JM: Grant and Mark, welcome to Software Engineering Daily.

[00:03:42] GM: Thank you so much for having us.

[00:03:44] JM: You guys have been building Replicated for about five years and your company requires you to have a detailed understanding of how enterprises are buying software. Describe to me the modern large enterprise and how it buys software.

[00:04:01] GM: Yeah, that's a great question. I think there's a lot of context that we can unpack there. First, I think that specifically for replicated, we think about software really needing to be enterprise-ready. We actually built this guide like three or four years ago called enterpriseready.io, and that guide sort of underlines how SaaS and software companies should be building software that enterprises will want to buy.

Turns out there're sort of this standard set of common features that every enterprise software application really needs to have. I think for a long time, people didn't really – They kind of thought about those features as one-off features, or maybe sometimes they were special requests. But ultimately over the last few years, it's really consolidated down with these like core features, and this are things like single sign-on, audit logging, role-based access control, change management, product security, team management, integrations, reporting.

Those features need to be present in the software that an enterprise is going to buy. From there, obviously they need to have like core business value, but that's the thing that really differentiates how an enterprise is going to evaluate a product versus how an SMB or midmarket company might be evaluating.

[00:05:22] JM: There's a shift from the top-down purchasing process that might be led by a CSO or CIO. That's the historical form of purchasing, but now there's more of a bottoms-up purchasing process that can often involve engineers. Maybe the engineer buys a cloud service or an API and starts using it in their product, or they start using the free version and then eventually many people are using the free version within the company and then eventually there's an enterprise deal. That's more of the bottoms-up approach.

At an enterprise, how does the breakdown between different software products distribute between bottoms-up and top-down? Is there more bottoms-up for top-down software distribution in purchasing?

[00:06:07] GM: Yeah. I think you're right. This is definitely changing and the world is shifting more and more towards this sort of like bottom-up model, and that's happening for a handful of different reasons. The first part you mentioned, like just the availability of cloud services and software tools that folks can just start adapting. I'd also say that open source has really been a huge factor in that as well. It's just not that hard for someone to find an open source project and included in a product that they're building and then for that to become a core part of the infrastructure.

You have these sort of like empowered developers and empowered sort of frontline folks who are finding tools and projects that really help them do their work. As they start to use those, often times these are freemium, or low-cost, or open source, and so they lack a lot of these like enterprise-ready features that we talked about.

As those applications and those projects become more widely adapted in an enterprise, you start to see the demand for more controls and more functionality and more features. As that happens, you start to get this like – Starts with one or two teams and then maybe it spreads out to 10 teams. There's this acknowledgment within the enterprise that, “Okay. We need to take this new tool that's starting to become really important in our workflows, in our processes and make sure that it's enterprise-ready.”

That's often the time when that sort of bottom-up motion starts to really become sort of a combination of top-down. They’ll engage a sales team and they’ll start to talk about the sort of good, better, best pricing plans. They’ll move from that sort of easy to adapt freemium model to the next highest plan or maybe a plan after that and maybe they’ll talk about spreading it throughout the entire organization. An enterprise could look at a piece of software and say, “Wow! There's a lot of success that these 10 teams are getting from this software. If we got our entire organization to use these and we created like an enterprise license agreement, it could really be an advantage for us.”

They’ll start to negotiate what that looks like. Then again, they’ll be asking for these features around security and scalability and how do they integrate it with all the other software. That’s when a salesperson or like a head of product, someone's going to step in and kind of help lay the roadmap for how all that's going to work together.

[00:08:46] JM: I think many of the people that listen to this show are either working in enterprise or they’re working on a software vendor that sells into an enterprise. Maybe they work on a database, or an authentication solution, or a security product, or a log management product. These software vendors, some of them are going to be selling purely cloud products. They’re going to be selling purely maybe through the cloud marketplaces. But many of them will eventually get to the point where they're going to need to sell to a wider array of customers, including large enterprises with a variety of servers.

I’d like to know from your perspective, what are the biggest struggles that I'm going to encounter if I'm a software vendor as I am going to market. If I want to actually sell a software product into an enterprise, what are the biggest struggles that I'm going to encounter?

[00:09:52] GM: Yeah. I mean, first of all, there's like all of the standard go-to-market challenges, right? How do you get awareness? How do you get people to want to buy? How do you create urgency? But then on the actual development side, I think that's where, one, you’re going to have to develop these features, like these enterprise-ready features.

Then one of those core features I think you're kind of alluding to is this idea that a multitenant SaaS application is often times like not very tenable inside of these large organizations, right? The data that an AI application or a developer tool might require could be seen as very sensitive information or sensitive data.

That large enterprise might start to ask for we call like an on-prem version of that product, right? This kind of goes by many names; self-manage, self-hosted, but a version of that software that isn't hosted and managed on servers that the software vendor controls, but rather is deployed to servers that the enterprise controls.

Now, those servers could be racked and stacked in a data center somewhere or they could actually be in AWS or in Google or in Azure, but just in the enterprise’s account, right? These are controlled from like a logical level by the enterprise. This is a really complicated problem to solve. It's always been sort of in software over the last 20 years. There's been this like – It's like almost this religious belief that's like you can't do both. You either are a SaaS company, like a multitenant SaaS company, or you are like an on-prem software company.

That's because like there're just a lot of challenges about doing either, right? If you're doing on-prem software, you have to think about releases and disconnected supportability and how do you make sure that you’re keeping your customers up-to-date, rather than having them all in disparate versions. There're all of these really interesting challenges that come from that. Plus, just the fact that when you think about 10 years ago, building reliable software was just so hard right and there was all these different sort of tools and ways that people going about it.

Ultimately, I just don't think that we had the right patterns and primitives for building truly reliable software, but that's all changed, right?

What we've seen in the last five years is that almost all software development is moving to this sort of like cloud native architecture. Really, in the last two years, we've seen just the domination of Kubernetes. Kubernetes in our opinion is truly the canonical patterns and primitives, building reliable distributed software. With that, it allows a software vendor to use the exact same deployment model that they're using to deploy a multitenant SaaS application to reuse that and deliver that manifest plus those images that sort of create that Kubernetes application and deliver that privately to enterprises to run their own version.

There's a whole bunch of other interesting challenges once you start doing that, but generally that sort of model we think creates what we call sort of modern on-prem and this idea that you as a software vendor, you don't need to fork your codebase. You don't need to think about it as a separate product. You need to just leverage these new patterns and primitives and deliver a Kubernetes-based application to your customers so they can run it privately. That's not for every customer. That's going to be for the largest enterprises who are really concerned about the security of the data that they want your application to process.

[00:13:51] JM: Can you describe in more detail why and how Kubernetes has changed the relationship between the enterprise buyer and the software vendor?

[00:14:02] MC: Yeah, I think there's a couple of different things that it's changed there. One is instead of every time the enterprise receives software, they have to understand how to deploy it. What are the requirements? What resources do I need to provision for this and what steps do I need? What one-off process inside my organization in order to get this software up and running and monitor and operate it? Now it becomes, “Oh! This is a Kubernetes piece of software. So I can apply all of the practices, all the knowledge, all the skills that we've been building inside our enterprise so I can deploy this thing. I can monitor it using the tools that I already have monitoring my Kubernetes infrastructure.”

Then on top of that is that the application will conform to those Kubernetes primitives around these common interfaces that Kubernetes provides around networking, storage, container runtimes, interfaces like this. If the application developer has written – Traditionally, an application developer will write code and say, "It needs this support and it expects this volume – This directory is hardcoded inside the code, whatever it is."

All these assumptions they make about the runtime are now configured through that interface. The enterprise, when they're operating it and when they're installing it, they can choose to relocate that. They can choose to create a service abstraction on top of it. They can apply a service mesh to it. They can run the storage however they want to. It allows them more flexibility for that last mile configuration of the application that historically was just tightly coupled into the application.

[00:15:25] GM: Yeah. I think it really becomes the common substrate for both vendors and enterprises to build and deploy software. This has tons of benefits beyond just like portability, right? We're also talking about the fact that developers, their skills can now translate from company to company because as everyone starts to use this canonical stack, you're not entering into Facebook and trying to learn their system of container deployment, which they call Tupperware, or going to Twitter to do Mesos, or doing like everything is moving to Kubernetes. That means that, as developers, our skillsets can really transfer and we can walk into a new company, pickup their stack, know sort of like the patterns that they're using because we've seen this many times before.

[SPONSOR MESSAGE]

[00:16:24] JM: Today's sponsor is Datadog, a scalable, monitoring and analytics platform that unifies metrics, logs, traces and more. Use Datadog's advanced features to monitor and manage SLO performance in real-time. Visualize all your SLO's in one place. Easily search, filter and sort SLO's and share key information with detailed intuitive dashboards. Plus, Datadog automatically calculates and displays your error budget so that you can see your progress at a glance.

Go to softwareengineeringdaily.com/datadog and sign up for a free 14-day trial, and you will also get a complementary t-shirt from Datadog. Just go to softwareengineeringdaily.com/ Datadog. Sign-up and get that free t-shirt.

[INTERVIEW CONTINUED]

[00:17:22] JM: The first idea that you're talking about there is that the fact that Kubernetes has become the canonical way of managing infrastructure means that as you go as an employee from enterprise to enterprise, you're going to have a consistent experience of how you're managing your infrastructure as opposed to the status quo today or perhaps certainly two, three, four years ago where you had to learn whatever ad hoc infrastructure solution the company has built to manage their different servers, whether they're doing some kind of chef situation, or they've just got a bunch of servers that they manually run scripts on. Some kind of distributed in-home, in-house bash solution that they've built.

Today, many, many, many companies are moving towards Kubernetes as the unified way that they're going to be managing their infrastructure. Then a downstream impact of that is that if you want to sell software to these companies, you now can have some expectation of the kind of medium that your software is going to be deployed on to. That simplifies the sales process for distributed applications, because most of the important software that is getting sold to enterprise today is some kind of distributed system, some kind of database, or logging solution, or something like that that is sold that it has to be distributed. It has to be managing multiple servers, multiple containers.

In the past, that was just a very hard solution to sell, because it was going to be – The deployment medium was going to vary from company to company and the management of different servers was going to vary from company to company. If you have a steady infrastructure medium, then you as a software vendor can have an expectation of how you want to deploy a distributed system.

Then I think one other benefit is if you have an expectation there, it becomes much easier for you as a vendor to work with the enterprise that you have sold to on an ongoing basis, because usually when you sell software to them, there's going to be some kind of maintenance agreement. There's going to be some kind of ongoing solution and you're going to have to do a little bit of work with the company.

If you are a vendor and you've sold software to an enterprise, how is that consistency of Kubernetes making your life doing maintenance easier?

[00:19:58] GM: Yeah, that's a great question. I mean, part of it is because the underlying platform of Kubernetes provides so many important reliability primitives that you sort of know that your application is going to be able to scale horizontally and really have a lot of the other sort of like core things that you want to have an expectation for how your application can run successfully, right?

You're getting all of these consistent patterns and primitives. You know that it's going to work there. If they can run Kubernetes applications – One of the things that we try to provide are like what we call preflight checks. We're going to talk about this a little bit more later.

But you can validate that the conformance of their cluster meets the requirements of your application. You're basically able to say, "Hey, look. Here's this common pattern we both follow to deploy scalable applications. Here's our manifest that describes how this application should be run and we are –" It's almost like we're taking with this manual operations concept where I used to SSH into a server and make configuration changes. Now, instead, we're just writing these manifests and then we're handing that manifest to Kubernetes and we're letting Kubernetes manage the application and the underlying infrastructure to conform to the desired state that we described in that manifest.

If I'm handing that to my servers that are in AWS, why can't my enterprise customers just hand that to their servers that are in AWS have that same desired state applied and run the application just as scalably, just as successfully?

[00:21:36] JM: Tell me more about the typical operations that a vendor might want to help an enterprise with. So if I've sold a piece of software to the enterprise, what does my ongoing life look like? I know it's not just a one-time self-service sales. What kinds of work am I going to have to do with my enterprise customer as time goes on?

[00:22:00] GM: Yeah. I mean, there's a handful of things, right? Number one, you have to think beyond just day one of like deliver the software and get it running and you really have to think about day two and sort of this long-term operations of the software. That's your number one. How do you make sure that you're delivering updates in a way that the enterprise can process those and deploy them through the same processes they used to deploy their internally developed software? That's a core concept that we believe in.

The other part would just be things are – We don't think that the world is perfect by any means yet. I think that like Kubernetes applications, like this is an error-free way. There's probably going to be issues. So you're going to need some type of troubleshooting that you can do long-term and do that in a disconnected way where you're not taking control of the machine remotely. You're trying to do something where you're still getting access to all the data.

We've built frameworks around this. We call it Replicated troubleshoot, which is a framework for identifying different log files and commands that you might want to run in your customer's environment to identify potential issues and then run it through redaction tools and analyzers that will sort of supply next actions the customer can take to troubleshoot the installation or some issue that might be happening.

There're a handful of other things I think you're going to need to be doing. Part of that is, is making sure that your updates are continuously successful. As you add new components, as you add new parts to your software, you need to make sure that the environment is able to manage and handle those. Finding ways to validate that every time you deliver is also really important.

[00:23:52] JM: The reason we're discussing this is because, at Replicated, you have been focusing on software delivery, the software delivery process that improves things for the vendors and improves things for the enterprises that are consuming it from the vendors. Describe the stack of software that you've built to enable that software delivery process.

[00:24:15] GM: Yeah, sure. I'll give a quick overview and I'll let Marc go into a little more detail. But we basically have developed a suite of open source projects, and these are really purpose built Kubernetes projects that are designed to sort of help facilitate the common challenges around delivering and managing Kubernetes off-the-shelf software.

We've open sourced all of these, and so that was a really – That's a big part of what we do. We call it Kubernetes off-the-shelf software. We refer to it as KOTS, and this is kind of a play on traditional commercial off-the-shelf software which is called KOTS. It's kind of how the industry would talk about shrink wrap your package software 15 years ago. We're kind of revitalizing that term for the Kubernetes era.

We have these different KOTS sort of components that are all open source and really tackle different challenges. I'll kind of let Marc dive into those.

[00:25:08] MC: Yeah. At the core of it, we have this – The main KOTS product is a CLI. It's written as a Kub control plug-in. It executes on the client side, and we made the decision just so that we could work with existing clusters really well without having to ask for higher security requirements or like cluster admin level permissions.

With that Kub control plug-in, an enterprise can just receive the upstream URL of an application in a license file and then install it. when it installs, it actually installs the next component that we have, which is called – We call it KOTS ADM, and that's the in-cluster admin console. That's presented as a UI that sort of inside the cluster and it has a few core functionalities like a dashboard so that you can add some Prometheus metrics in like the health of the application in general. View the license, sync the license if you've requested changes or the expiration has changed on it and then view the downstream – Sorry. Vie the version history that are available in all the downstream clusters that you have it deploy to so you can see the diffs that are applying. You can check for updates. You can make those last mile customization changes.

One of the things that we built core in the KOTS and KOTS ADM from the beginning is a deep integration with a project in the Kubernetes ecosystem called Kustomize, with a K, and that allows the enterprise to take the YAML and make last mile changes to it that are relevant to their configuration, but probably not relevant to other configurations and other clusters that need to run it.

Then we also have built support with preflight checks and troubleshooting that Grant mentioned earlier. If the application stops working, you can click a button and it'll collect a bunch of support bundle type data collected into an archive, redact it and then even perform automated analysis against that to show you what might be wrong based on what the application vendor has packaged into the application and a lot of the stuff that we've done.

Then the last part to mention is that it enables you to kind of move away from this click to deploy, like my application received an update, so you click a button and it allows you to like integrate this into a GitOps workflow so you can connect your repository, whether that's GitHub, GitHut Enterprise, Bitbucket, GitLab, wherever that is. Then updates to the application, whether they're license updates, configuration changes, customization changes you've made or the vendor itself has updated the application. They'll just be created as commits into that git repository, which then you can run that through any kind of workflow you want for security scanning or any type of like post or pre-deployment validation or those manifest before they get deployed to the cluster.

[00:27:43] GM: Yeah, I think that part is really important, right? I think last summer on this show, we talked about a project that we call Replicated Ship, and KOTS ADM is really the spiritual successor to Replicated Ship. We sort of took what we learned building sort of Replicated classic, plus Replicated Ship, and we created this KOTS suite of tools, right?

One of the core concepts of ship was both like getting customized and that sort of last mile configuration, but it was also we really believe in this concept of GitOps, right? The folks at Weaveworks kind of have pioneered this idea, but GitOps is really about version controlling all of the sort of configuration as code. All the manifest, before you actually deploy those to a cluster, you should check those in the version control in order to use the change management processes that we're also familiar with with version control.

From our perspective, the really cool thing this does is if you think about like how you manage third-party software versioning, like for long time, you might be doing like – In the Kubernetes world, you might have done like a Help upgrade or a Helm install, like these sort of different like commands that are just going to operate directly in the cluster.

Now, with what we've done with KOTS and KOTS ADM, you can actually manage third-party software versioning through version control. We think that just makes sense and it's automated. You don't have to do anything extra. Anytime the upstream application ships a new update, we’re going to automatically download those manifests. Take those images, push those into internal registries that the enterprise might have. Rewrite the images on top of the manifests and then commit that directly into the version control system that the enterprise has set up.

From there, it can flow through their internal processes around policy enforcement, image scanning, whatever else you want to do before it gets deployed. With those deployments, then have this really clear record of all the versions that have ever been delivered to the cluster.

[00:29:48] MC: Yeah. I think one of our goals with that, the main reason we want to do that was to be able to receive third-party software using the same pipelines that you’re already using to deploy your first party software. If your team is writing APIs and deploying them as part of your product and you're also receiving third-party software you need to deploy to that cluster, there shouldn't be two different processes involved with getting those release. They should all work through the same pipeline because you've already invested in that. You already have the tooling and the visibility and everything into that.

[00:30:14] JM: This set of tools that you’ve built is about 5-1/2 years in the making, and I’d like to get an understanding of how you arrived at this particular stack and maybe some of the errors or blind alleys that you’ve found along the way. Why did you arrive on this set of open source tools, KOTS, KOTS ADM, preflight and so on as the set of solutions for being a software vendor that wants to deploy software to an enterprise or an enterprise that wants to consume software from a traditionally cloud vendor?

[00:30:56] GM: Yeah. Ultimately, there are many iterations in the making, right? I think that, one, you spend time working with software vendors. Our customers are folks like Hashicorp, and CircleCI, and TravisCI and the folks you’ve had on the show, like Snyk and Bugsnag and YogabyteDB all use Replicated as well, and they’ve been using Replicated classic for years.

We work with 50% of the Fortune 100 to deliver applications like those I mentioned into these enterprise environments in totally secure ways. If some of these clusters are air gapped inside of a truck in the back of the Middle East, like you have to be able to do this in a very disconnected, secure way in order to support the most demanding enterprises. Our tools have really been designed from the ground up to do that.

Then the other part is we've just seen Kubernetes become – This project is so widely adapted and it just has such velocity. For a long time – We actually wrote our own orchestration and scheduling when we first started the company. Our goal there was not like to be an orchestration scheduling company. It was just like fill some of the gaps that we've felt like were missing to deliver software this way.

But then, maybe three years ago, we delivered a version of Replicated that was Kubernetes compatible. This version that we've delivered is the first truly Kubernetes native way to deliver third-party software. I say that because it really meets the entire spectrum of end customer needs, right? Online or air gap, we can solve both. It also does something what we talk about as existing cluster installations or embedded cluster installations. I'll differentiate those really quick.

Some enterprises have an existing Kubernetes cluster and they need to install your application in it. May they'll create a namespace for you that they want to deploy your software to, but you need Kubernetes native tooling to make that happen. That's where Mark kind of mentioned this idea of KubCTL plug-ins, but these KubCTL plug-ins are used by the cluster operator. KubCTL, anyone that has Kubernetes like has KubCTL on their workstation or on some machine they use to control the cluster. Extending KubCTL with functionality, like preflight, or troubleshoot, or KOTS really gives native experience to the end user to manage that cluster.

Then the other side is there are some organizations that enterprises don't have really any idea what Kubernetes is yet. That's starting to become less and less, but there's still probably 50% of the market that doesn't have any kind of Kubernetes initiative or cluster that they're running yet. When you deliver them your software, you need to package up or embed Kubernetes with the application. We open sourced are embedded Kubernetes installer that we call KURL. This is your communities URL, and it's basically a Kubernetes distro creator that can package up some of the core components of a Kubernetes cluster, because raw Kubernetes in KUB ADM doesn't provide all of the like necessary components or add-ons that you would need to create like a usable cluster.

KURL sort of is built on top of Kub ADM. It adds in these different add-ons and it creates like a very usable cluster that your application can be deployed on to. That can just be installed either from an air gap package that we generate or from like a URL, like a KURL [inaudible 00:34:24] URL that has like a little hash at the end that is an installer for your application.

Again, this is totally open source. We open source that. We open-source to all these tools because we think, number one, that anything with the cluster operator is running, that's going to power the installation and the management of one of these third-party software applications. It should be open source, rate? It just makes it more transparent. It’s easier to adapt. Ultimately, we've figured out a business model where our customers are the software vendors. So we have a commercial product that helps them sort of operationalize and scale the distribution of all this, right?

We talk about that towards the end, but like we don't need to monetize or to keep these cluster operator tools proprietary. So we open-source them, and they’re really focused on this idea of sort of optimizing the experience and automating as much as you can and reducing the amount of human interaction in terms of supporting and updating and managing these applications.

We spent a lot of time with the existing toolset. Ultimately, we decided that these are the core features that we needed have, and these are the core features that any like Kubernetes application that's been delivered to an enterprise should have. Any third-party software application delivered through Kubernetes needs some way to be configured and managed, then a way to troubleshoot it and ensure that it's going to run correctly on every update. Creating these frameworks and these tools is our way of sort of helping the industry move towards this idea of modern on-prem software.

[00:36:06] JM: Let's go through some of these open source building blocks that you have. One of the projects you have is called KOTS, which is Kubernetes off-the-shelf. Can you explain what that is and why you needed to build a Kubernetes distribution?

[00:36:22] MC: Sure. KOTS is not a Kubernetes distribution by itself. It's a tool that's designed to deploy a commercial off-the-shelf software and help you operate commercial off-the-shelf software inside any Kubernetes cluster, and that can be a Kubernetes cluster that was embedded in with the application. It can be a cluster running on Amazon EKS or GKE or Azure, or it can be open shift. It can be whatever you want, air gap, non-air-gap. The challenge is that application vendors were shipping their application in various ways. Helm charts are a very popular way to distribute applications. Other application vendors would just write application YAML and deliver that to you and then you'd have to confi gure that more.

There was still a little bit of disparity in how you would configure and manage that. Then enterprises needed to inevitably make last mile changes that were only applicable to their enterprise, to their cluster, to their installation. KOTS is really a tool that tries to help normalize all that through a Kub control plug-in so you can operate this from your workstation. Point it to whatever cluster that your Kub config your Kub context is already set to.

You can say Kubctl KOTS install, whatever application it is, and that'll install the admin console. Get it ready for a license and get the application ready to go, from which point you can just take it and run with it and build it into any workflow that you might already have.

[00:37:46] JM: Got it. Sorry for the confusion there. So KOTS ADM is a sidecar tool for managing the applications that you create with KOTS. Can you describe what KOTS ADM does?

[00:38:03] MC: Sure. Yeah. It's exactly the way we like to talk about it, right? It's a sidecar application. Think of it like an admin console for the application. It's optional. But when it's there, it provides a lot of value around a dashboard and visibility into your license. The ability to troubleshoot it, it creates a web UI around prefl ight checks so that you can see if your cluster meets the minimum requirements or has suggested changes that you need to make or upgrades you should make to it.

Then the ability to visually see the diff. Kubernetes manifests are becoming that common substrate that we talked about. It allows you to visually see the diff and the cluster operator can click in when there's an update available and even see the YAML lines of the Kubernetes application that have changed with that update and then they can decide whether or not they want to deploy that, or they can then, as Grant mentioned earlier, integrate it into their existing GitOps pipeline if they have that and be able to configure it to just push all those updates to their Git repo.

But KOTS ADM is a sidecar admin console that will deploy alongside an application. It's a pretty small, not very resource-intensive, but it provides a lot of admin functionality that every application developer would probably have to create through like CLI's or their own little web UI.

[00:39:16] JM: Okay. I think the KOTS and KOTS ADM of the six different open source projects that you have, if we want to talk through the process of using your suite of technologies as a vendor, the main idea is that if I have some kind of software that I want to deploy to an enterprise as Kubernetes cluster.

I'm a vendor. I'm going to make my software compliant with Kubernetes off-the-shelf and I'm going to be able to deploy that to the enterprise as Kubernetes cluster, like if I've got my database, like if I'm YugabyteDB, I create my database. I make it possible to deploy to the Kubernetes off-the-shelf version and KOTS will let me deploy it to that kind of Kubernetes, and KOTS ADM will let me remotely manage aspects of my vendor distribution. KOTS ADM kind of provides the medium of creating updates and doing kinds of operations, other administrative duties and creates a medium of communication between the vendor and the enterprise that's actually consuming the vendor software.

[00:40:37] GM: Yeah. I think that's pretty close. The key piece there is that KOTS ADM lives inside of the enterprises cluster and it becomes this like tool that automates some of the toil that the enterprise IT admin might be doing in order to manage updates or to troubleshoot the application. It's being deployed along with the application. It lives in the cluster. There's no like remote management as much as it can make an outbound request to find updates that you as the software vendor have published. Then it can also extract logs and other information, redact those and the bundle those up either for in-cluster analysis or to be delivered for the vendor for remote or for disconnected troubleshooting.

We should think about KOTS ADM as this like really powerful in-cluster sidecar component for any software vendor that's delivering a Kubernetes application. Then I think probably the most important part – So like out-of-the-box, the end customer is going to do this sort of – They're going to put in their configuration information around like, "Oh! What's my hostname? Where are my TLS certs? How do I configure this app?" That's very similar to like Helm values.

But then the important part is that once they've set up the application and have it running, they can actually set up the day two operational automation, and that's the stuff that's going to push images into the enterprise’s internal version control system and their internal image registry.

That automated process means that, now, instead of like having a task on my list every two weeks, go check for an update and download the software and run it through some scanning and then like deploy it out, I can basically just use my version control system to merge in these commits to my production deployment, and that is sort of will run through all of my standard processes that I'm using to deploy other Kubernetes services that my internal software teams are deploying. It's piggybacking on that same process it used deploy internal software and saying, “Look, now, a third-party software can be managed the same way in a very automated process.”

[00:42:50] JM: Take me inside the engineering at Replicated. You've got these open-source projects. You also have the previous version of Replicated Ship and you kind of have the evolution of the replicated platform to support. I just like to know how engineering at the company is organized and what your process of communicating with customers and iterating on the product is.

[00:43:20] MC: Sure. I mean, it's a great question. We have three different versions of Replicated that we have out there that we’re supporting right now, and they weren't ever a complete rewrite from the ground-up. We would continue to build on all of the tooling that we had built in the previous iteration. Even our Kubernetes distribution, this KURL project that allows you to create a one line Kubernetes distribution is really just an open sourced version of the Kubernetes install scripts that we've been running for the past 3+ years for somebody who had a Kubernetes version of Replicated classic.

We've worked really hard on the engineering side to make sure that we don't have three totally different sets of codebases in three different product lines that we’re actually offering. They’re all built on the same thing and we’re actually working hard to take the Replicated classic code right now and have that – Instead of having it have a fork of the open source stuff that we've created and released under the KOTS name just to be able to use that. They’re all using the same common code base internally.

[00:44:24] GM: That's because we're a Go shop, right? A lot of times, people vendor these in.

[00:44:28] MC: Yeah, exactly. I mean, almost all of our code on the backend is written in Go. All the open-source stuff is in Go. They reuse each other a lot. A lot of the things that we do are around Kubernetes operators and Kubernetes custom resource definitions also. We have these custom kinds that we've created inside our different projects. So when we want to use them, as Grant mentioned, we vendor them in and we just create custom resources that are able to use each other.

[00:44:53] JM: What's on the roadmap. What is important to implement in order to improve the process of software delivery from your point of view?

[00:45:03] GM: Yeah. I think the interesting piece is we've been working towards the Replicated KOTS offering. I mean, really, for five years, right? It's sort of over several iterations. I think maybe three years ago when we started to really make Kubernetes a first-class part of what we're doing, we had some of these ideas. It's been a long road to sort of find the right way to offer this. I mean, partially because the ecosystem is moving very fast, right?

If you think about this idea of Kubctl plug-ins, the whole framework was rewritten maybe like nine months ago or something and we think that's a really important part of how cluster operators should be managing these third-party components. As everything has been changing, we've been trying to make sure we're taking advantage of the latest and greatest different ways to offer this functionality.

This project and in these suite of tools is really exactly what we wanted to have for a long time. I think that it's an incredibly strong foundation for solving some of the most important problems around configuration, troubleshooting, updating, automating away this toil.

When I look at the roadmap, it's continued to integrate these projects with the broader Kubernetes ecosystem. Our KURL project, that's the open source sort of Kubernetes distro creator. We're going to integrate that more closely with the add-ons project. That's our sig that's happening in the Kubernetes ecosystem. Integrate it more closely with the cluster API. There're some really interesting things we can do there. But ultimately our goal with that project is to continue to commoditize the installation and update and management of just a raw Kubernetes cluster. We don't think that should be some special and hard thing to do. We just want to see that so that everyone can have a cluster no matter where they are.

Then these other projects, I think about the ability to do preflight checks and to troubleshoot these applications. We want to make sure that that project is well-adapted and the standard – And become the standard for application level troubleshooting in the Kubernetes ecosystem. We're trying to work with some of the folks at Mesosphere on the Kudo Project, and we just like to see – We would like to collaborate with more folks on these tools and really build community and consistency so that these are not just one-off tools that we created, but they become – And we migrate them toward standards that we all sort of agree will create a better ecosystem for delivering these types of applications.

I'm sure Marc has some other thoughts on roadmap as well.

[00:47:54] MC: Yeah. I mean, I think are our goal with KOTS is to make it so that there's compelling enough reason and it's integrated enough into the enterprise's existing systems and process that they have that whether or not the application is being delivered through a replicated vendor, it's just a good way to run third-party software, because you have guaranteed success that it's going to install. It's going to update. It's going to be compatible with all the workfl ows and everything that you want. We just really want KOTS to become an open source good tool that you can use to run any third party software inside a Kubernetes cluster.

[00:48:28] GM: It could Helm charts. It could be raw Kubernetes manifests. It could be operators. We think all of these can benefi t from some of the core principles around GitOps and automating workfl ows and having a consolidated dashboard for all your third-party applications. That KOTS ADM product actually has a really cool feature, which if you use it as a software vendor to distribute your application, your customer can use it to manage that one application and it's totally white label and it looks like your admin console.

But then overtime as that operator really loves these patterns around having their third-party application updates automated through version control and in their internal registries and they're not doing the work on that, they can actually start to add additional applications to that admin console and it becomes multi-app. Just like kind of how Slack can go from like one Slack workspace, to 2, to 3, to 5. This tool use the similar UI and experience to allow you to manage multiple applications as a cluster operator in this very automated way.

[SPONSOR MESSAGE]

[00:49:41] JM: As businesses become more integrated with their software than ever before, it has become possible to understand the business more clearly through monitoring, logging and advanced data visibility. Sumo Logic is a continuous intelligence platform that builds tools for operations, security and cloud native infrastructure. The company has studied thousands of businesses to get an understanding of modern continuous intelligence and then compile that information into the continuous intelligence report, which is available at softwareengineeringdaily.com/sumologic.

The Sumo Logic continuous intelligence report contains statistics about the modern world of infrastructure. Here are some statistics I found particularly useful; 64% of the businesses in the survey were entirely on Amazon Web Services, which was vastly more than any other cloud provider, or multi-cloud, or on-prem deployment. That's a lot of infrastructure on AWS.

Another factoid I found was that a typical enterprise uses 15 AWS services, and one in three enterprises uses AWS Lambda. It appears serverless is catching on. There are lots of other fascinating statistics in the continuous intelligence report, including information on database adaption, Kubernetes and web server popularity.

Go to softwareengineeringdaily.com/sumologic and download the continuous intelligence report today. Thank you to Sumo Logic for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:51:31] JM: Could you tell me more about the vision for these open source tools actually developing into an ecosystem? Because you're hinting there the reason that you had to go from Replicated Ship or you chose to go from Replicated Ship to this suite of open source tools. The idea there being that a company like Mesosphere and perhaps other companies in the future would want to use some of these different tools to help facilitate the software delivery process or things that are components of the software delivery process.

I'm just trying to understand your vision for the future of how software is delivered. How software is purchased? The process of deploying it on to Kubernetes clusters. Clearly, you are envisioning a very large potential market because you're indicating that the market is so big that it would not only support multiple commercial players that are going to want to do stuff like this, but so big that these different commercial players are going to want to be contributing to shared open source resources.

I'm just curious about your perspective for how this market is going to develop over time.

[00:52:51] GM: Yeah. I mean, ultimately, we really think that the world of enterprise software, like it's always been this sort of duality of on-prem versus SaaS. Our vision of that, these two worlds should really blend into one. We think that Kubernetes is the primary driver for that blending. Ultimately, the idea that like every company is going to run a thousand different SaaS applications and send data off to all these different multitenant apps, we think that's insanity, right? We just don't think that the large organizations are going to want to expose their data to that many different vendors. We don't think that that's a secure way to think about your data.

You as an enterprise are really a steward of so much data and you have consumers that trust you. You have other businesses that trust you. When you start to just like willy-nilly toss that data into any vendor or you let anyone sort of bottom-up use these applications without the considerations around their compliance and their security, even if you're trying to do vendor security questionnaires for every piece of software that you use – I mean, trying to manage the security and surface area of a thousand different software vendors, it's really hard. You just have so many different points of attack.

We see a world where you have fewer trusted and vetted software like infrastructure providers, and those could be the big three hyper clouds, like Amazon, and Google, and Microsoft where you're using there as a service, but it's really just infrastructure as a service. Maybe you have a handful of other SaaS applications that you also use. Maybe it's Cloudflare, maybe it's

Ssalesforce, whoever else, but the number is definitely not zero vendors that you trust with your data. The number is definitely not a thousand in terms of vendors that you trust with your data. That number has to be somewhere in the middle, but we think that the world of software, like you have to be able to use a lot of software.

The only truly secure way to use software in our opinion, is to deploy it where you have control over the data that you put into it. You’re not exposing any of this additional data. If you can use lots of third-party applications deployed into servers that you control, well, that's sort of like the panacea. That's perfection in our opinion.

If you could automate the process around how those applications are actually like operationalized and how they're updated and how they’re managed and they really become self-healing, well, then you've just – Like I said, “Okay, now I don't have to think about the data security around these applications nearly as much, because it’s the same data security that I use for internally developed and deployed applications.” I don't need to think about the operational overhead, because the same operational patterns that I used to deploy internally written applications.

We can start to expose more and more software applications that your team can run in your infrastructure and they can start to take advantage of lots of different tools and automate more processes and really achieve that like concept of like overused term of digital transformation and become a software company.

That vision I think is it's so enormous because you're talking about taking a market of multitenant SaaS applications. It's about 100 billion a year and adding it to a market of on-prem software deployment, which is still about 400 billion a year, blending those together in a way that accelerates the adaption of software.

We just think that this model –And when you look at all the alternatives, you look at like enterprise key management, which is like a solution around how do you encrypt data, but keep the keys with the actual enterprise. These data still require trust. These models still require trust around the data. Maybe someday we get to a fully and superfast version [inaudible 00:57:09] encryption and it's not like a science project, but it's actually something that we are operationalizing and it makes sense to use for terabytes and terabytes worth of data. Everything we know about that doesn't seem to make that like really reality, at last not in the next 20 years. It’s not with our current computing technologies.

We think that this idea of modern on-prem is really part of the future and it's a core driver for how enterprises will make that digital transformation. With Kubernetes at the center, we just see it as this hugely – Like there's this huge amount of momentum behind it already and we think that it's clear that running third-party software in Kubernetes applications is like the best way to run third-party software.

No one should be delivering OVA's or JARS or WARS, and we see a lot of these like traditional software companies that had built their applications and have tens or thousands of different enterprises using that sort of like legacy on-prem software. They're all making the same transition as the SaaS companies to create Kubernetes versions of these applications, because that way they can take advantage of the patterns and primitives of reliability that Kubernetes provides and their customers can run their software much more reliably and reduce that operational overhead.

You’re seeing these markets really converge into one way of delivering what we call Kubernetes off-the-shelf software. For us, we want to provide the set of tools that help spur that forward and we are thrilled to work with anyone who sees that same vision and wants to create tools to make that more possible. We think the success of Kubernetes overall, like when you step back and you compare it to Docker or something else, we think it's because it was so open. It's because it was community owned and community-driven. That's why we’re thrilled to work with anyone that's making these – That wants to see this reality come true, because ultimately standards are really important and they create the longevity and the opportunity behind how something becomes not just the five-year opportunity, but a 20 or a 50-year opportunity.

[00:59:33] JM: I agree with what you said, and it's hard to know exactly how the market will shake out. But it's certainly conceivable to me that it is as big and it’s going to be changing as much in the direction of Kubernetes as you’re indicating. But just to give a little bit more color on the particular area of software that you are solving, that is the delivery process of a vendor, delivering to an enterprise.

That process actually has a number of roles involved and it's not just like I am a vendor and I am going to deliver the software to my enterprise and it's going to all be handled by a solutions architect. There's actually – Multiple different people there are going to be involved. There's going to be customer success. There's going to be sales. There's going to be engineering, solutions architecture and all these different roles within the software vendor, within the company that is maybe the database, or the logging solution that's selling to the enterprise.

My understanding of one of the things you're trying to solve is the communication channel where all of these different roles within the software vendor are talking to the enterprise. Do I understand correctly that there are just like frustrations or difficulties around the communication channel between that vendor and the particular customer and you're trying to consolidate some of that communication?

[01:01:12] GM: Yeah. This is really part of our commercial offering, which is the vendor tooling to operationalize and scale the distribution of Kubernetes off-the-shelf applications, right? This is a product that our customers are using for years and it's really just focused on things like workfl ows and processes for release channels, licensing and entitlement managements. This is like how do you create a customer license? How do you create different values and enable different features per customer? How do you make sure you're delivering the right version to each customer and assigning them to different channels and then how do you manage the process around support and making sure that you have a clear process to troubleshoot those customer environments when an issue does arise?

To your point, we've made all these features be very cross-functional. The idea behind the commercial tooling is, "Hey, you're going to deliver Kubernetes applications, some images to a customer. If you going to do that like once or twice, maybe use some more open-source stuff. It's going to help you out. You can write these little manifests that will invoke the preflight and the troubleshooter or can make the KOTS ADM available.

But if you're going to do this and try to deliver your software to tens, hundreds, thousands of different enterprises, well that's when you need to find workflows and processes and that's what our commercial product really offers. That's when you would find that you're getting involved more than just an engineer to send some Kubernetes manifests, but you're pulling in, you got your customer success folks who want to know what version are they on. How are they using the software? What's the latest – When did they last check to see if there's new version?

Your support folks, you're going to want to manage the support bundles that are generated by each of these customers and then sent up once they're redacted. You're going to want to have folks that are in presale that can create a trial license, and maybe the trial license is generated from a Salesforce integration. Anytime you put a new lead in Salesforce, we generate a new license for them and you give your presales team the ability to walk in to a customer, give them a great demo, set up environment so they can try out your product, right?

There're all of these different folks that have to be involved in the process of going to market successfully with a Kubernetes off-the-shelf application. That's what the commercial replicated tools are really designed for.

[01:03:35] JM: To close off, what has being in the hardest part of building the business. Over the last 5- 1/2 years, what was the most difficult point?

[01:03:45] GM: I'll let Marc take a crack at first and then I'll give my perspective.

[01:03:49] MC: Okay. From an engineering and a product perspective, I think like the most difficult part is definitely been around just how fast moving in like the ecosystem is. Five years ago when we started the company, Docker containers were around, but Kubernetes didn't even have the first alpha release yet. So we needed to build a container scheduling and orchestration system to solve this.

Then Swarm came out, and if you think five years, it's not a long time to have gone from a proprietary scheduling orchestration system. Rewrite it so we could have support for Docker Swarm and support for Kubernetes, and then finally to where we are now where we like – It's Kubernetes native and it supports Kubernetes.

In general, I'd say one huge technical challenge in product challenges is just really been to staying ahead of and where we need to be to be compatible with both the vendor who often is bleeding edge and has Kubernetes application, has the latest and greatest and being able to still support enterprise customers who may be run disconnected offline older [inaudible 01:04:52] environments that they need to be able to make that all compatible in.

[01:04:56] GM: Yeah. I guess from my perspective, man, there are so many different challenges. I mean, this is the second business the Marc and I have run together. When we first started it, we felt like we had learned so much from the first business and now I look back five years ago and realize like how much we've learned now.

One of the core challenges that I've faced personally is just like how do you describe really complex and technical solutions that you're building that are super powerful. But like when the market doesn't necessarily – Like not everyone's up-to-date on all the technologies that you're talking about, but they're foundational in what you're doing. How do you describe that to folks in a way that they really can grok it quickly and see the value and want to learn more and dive deeper and read some docs? That's a really big challenge.

I think, particularly, it kind of couples on top of what Marc was talking about about how fast the market moves, right? With Kubernetes ecosystem, it's just been exploding. If you want to describe it, you have to reference the things that are enabling your technology, but you kind of need folks to be up-to-date. They have to be maybe on Hacker News every hour in order to know the latest and greatest or be following every release and reading every common blog posts that's out there. Those are real challenges around buildings this kind of business.

[01:06:14] JM: Guys, thanks for coming on Software Engineering Daily. It's been great talking to you.

[01:06:17] MC: Yeah. Thank you.

[01:06:18] GM: Thanks so much.

[END OF INTERVIEW]

[01:06:28] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[END]