EPISODE 768

[INTRODUCTION]

[00:00:00] JM: Cloud computing has been popular for less than 20 years. Large software companies have existed for much longer than that. If your company was started before the cloud became popular, you probably have a large data center on your company premises. The shorthand term for this software environment is on-prem. Deploying software to your own on-prem servers can be significantly different than deploying to remote servers in the cloud.

In the cloud, servers and resources are more standardized. It's often easier to find documentation and best practices for how to use cloud services. Many of the software vendors who got started in the last decade created their software in the cloud. For example, readme.io makes it easy for companies to create hosted documentation. Their early customers were startups and other cloud native companies, and all of those companies were happy to consume the software in the cloud.

As time went on, ReadMe found that other customers wanted to use the ReadMe product as a self-hosted on-prem service. ReadMe needed to figure out how to deploy their software easily to the on-prem environment. It turns out that this is a common problem. Software vendors who want to sell to on-prem enterprises must have a defined strategy for making those deployments to on-prem infrastructure, and those deployments are not always easy to configure.

Replicated is a company that allows cloud-based software companies to easily deploy to on-prem infrastructure. Grant Miller is the founder of Replicated and he joins the show to discuss on-prem, cloud, and the changing adoption patterns of enterprise software companies.

[SPONSOR MESSAGE]

[00:01:51] JM: HPE OneView is a foundation for building a software-defined data center. HPE OneView integrates compute, storage and networking resources across your data center and leverages a unified API to enable IT to manage infrastructure as code. Deploy infrastructure

© 2019 Software Engineering Daily

faster. Simplify lifecycle maintenance for your servers. Give IT the ability to deliver infrastructure to developers as a service, like the public cloud.

Go to softwareengineeringdaily.com/HPE to learn about how HPE OneView can improve your infrastructure operations. HPE OneView has easy integrations with Terraform, Kubernetes, Docker and more than 30 other infrastructure management tools. HPE OneView was recently named as CRN's Enterprise Software Product of the Year. To learn more about how HPE OneView can help you simplify your hybrid operations, go to softwareengineering daily.com/HPE to learn more and support Software Engineering Daily.

Thanks to HPE for being a sponsor of Software Engineering Daily. We appreciate the support.

[INTERVIEW]

[00:03:14] JM: Grant Miller, you are the CEO of Replicated. Welcome to Software Engineering Daily.

[00:03:18] GM: Thanks so much for having me.

[00:03:19] JM: On this show, we have talked about on-prem software many times. On-prem can mean many things. Typically means not in the cloud, and this will happen because a company usually gets started before cloud computing. So they already have servers. Sometimes it's a company that doesn't want to move to the cloud because they have sensitive information. There are plenty of reasons why a company might operate its own servers.

Describe how on-prem infrastructure affects the software development of a company.

[00:03:54] GM: Yeah, sure. So we actually have a slightly different definition of on-prem, and we call it modern on-prem. For us, what we mean by modern on-prem is basically we’re sort of defining the cloud as this combination of two things. The cloud would be both infrastructure as a service and software as a service.

So what we see modern on-prem is basically the ability for an enterprise end user to take a cloud native application, so something that could be deployed as SaaS, and instead deploying it into private resources that they control. So that could be a full on-prem data center where they actually racked and stacked the machines, or that could just be the AWS VPC that's owned by the enterprise.

[00:04:41] JM: How does that on-prem infrastructure affect software development? How do on-prem companies develop software relative to companies? Because I think a lot of the listeners are just used to building on the cloud. How does that experience compare to an on-prem company?

[00:04:57] GM: Yeah, sure. So the difference from our perspective really is around building something that's very portable. So interestingly, these worlds have really emerged a lot over the last, call it, 5 or 10 years, and the reason they've emerged is because the technologies that we use to deploy software have really shifted from this sort of like manual operations and batch scripts into systems like orchestration and scheduling platforms such as Kubernetes.

So from our perspective, a modern on-prem application is actually just one that has most of the operational knowledge sort of baked into its orchestration scheduling system and then becomes portable so that it can be installed into an Amazon VPC, into a Google VPC, or into on-prem data center, basically anywhere that there's programmable compute and storage on the other side.

[00:05:52] JM: Right. Okay. So let's talk a little bit more generally about on-prem infrastructure. So in cloud infrastructure, there are lots of random failures. So the software is orchestrated to be resilient to these random failures. On-prem infrastructure seems a little bit more different because it's for specific companies and you don't have to deal with this super high volume of bursty workloads. Well, the workloads just very. I mean, I guess it varies from company to company, but I just like to know a little bit more about the nature of the infrastructure at your average on-prem company. Do on-prem companies have issues with these kinds of failures at scale?

[00:06:36] GM: Again, it's kind of one of these things that's changed a lot over the last 5 or 10 years, and the reason it's changed is because the lines have really blurred between what is on-prem infrastructure and what is cloud infrastructure, right? So you've seen this recently with some of the cloud providers even releasing sort of on-prem versions of their infrastructure tools. So Google offers to GKE on-prem. Microsoft has had Azure Stacks for a while, and then you've seen things like Amazon released their –I think they call it Help Posts more recently, and that's an example of like an infrastructure sort of like backend becoming available to be deployed anywhere.

But even before that, like VMware compute was programmable, that was in the data center. So as compute became programmable in the data center, these sort of like – This fault toleration has really become much more of a global concept than it is just a cloud concept, right? I think even things like PKS, which is Pivotal Kubernetes Service, or before that, what they did with Cloud Foundry, taking a lot of the same concepts of cloud development deployment and bringing it to any environment.

Ultimately, the AWS infrastructure is on-prem for Amazon. Like Facebook, their infrastructure is on-prem for Facebook, google's infrastructures on-prem. For our perspective, it's just about who's actually controlling the software that's running on those servers, right? So when we think about modern on-prem, what we're talking about is the ability for an application vendor. You could think about it as a SaaS company, packaging up their application as a cloud native application. So that's using KUberntes and sort of these primitives that we've defined to create truly reliable applications and then delivering that to an enterprise who can install it into an on-prem data center, a VPC, and run a fully private instance of that application, but instead of having to operate the application so manually like they would've done 20 years ago, they're actually leveraging all of that automation and orchestration and scheduling that's baked in, something like Kubernetes manifests, to actually operate the application.

What you see now is the ability for instead of just like sending along the bits, you're actually sending the sort of like know how, the automation, to actually like deploy and scale and self-heal that application when it's running in the end-user environment.

© 2019 Software Engineering Daily

[00:09:04] JM: Now, you've mentioned with Kubernetes the importance of a platform as a service layer even at the on-prem companies. Although I guess Kubernetes is kind of a layer beneath that, but whether we're talking about Kubernetes, or OpenStack, or Cloud Foundry, or Mesosphere, or OpenShift, there are these software tools that give an on-prem infrastructure environments the feeling of being a cloud, where the operations team can set up and manage a cloud foundry or a Mesosphere and give instances, give server resources to that layer, and then the developers within the company can request resources from that layer, and it can feel to some extent like a cloud provider. So we do have this on-prem infrastructure that makes things feel like a cloud and we also have some on-prem companies, they're adopting the cloud. So it is obviously this heterogeneity of different on-prem, versus cloud, versus hybrid cloud deployments that we're seeing more and more of.

Tell me about the difficulties that some of these on-prem companies have when it comes to software installation and procurement. If I'm a developer at one of these on-prem companies, there are challenges to me getting whatever kind of software I want running on the infrastructure at my bank, or insurance company, or healthcare company, whatever company I work at that has this on-prem infrastructure. What are the challenges that I encounter as a developer when I'm trying to get whatever software I want to do the job on my on-prem infrastructure?

[00:10:54] GM: Yeah, sure. So we kind of talked about the evolution of enterprise software in this regard. So when you we look at what enterprise software meant 20 years ago, it was basically that some organization would like create these JAR files or WAR files and then distribute those to enterprise and customers. Then the end user, the enterprise IT admin, who's responsible for racking and stacking a machine, setting up the power supply, the operating system, everything else, runtime, databases, installing those package, all the dependencies and then keeping that all managed, and they were doing that in a very manual way. So like some system went down, your SQL server had a blip. Your DBA would SSH into that machine and like make some changes.

So that created this very operationally-intensive sort of like overhead for these enterprise IT admins, and so as a result, they weren't running a whole lot of enterprise software applications. Maybe you had your ERP and your CRM and your exchange server, but like that was about it, because you had to have IT teams schedule around the clock and sort of like follow the sun

support in order to make sure that those system stayed up, because they were so manual, right?

So what you saw when Benioff introduced the SaaS model, is that that really took that operational overhead and instead of every enterprise IT work repeating the same work to keep the services up, it centralized that work to the application provider, the SaaS company. So that model became pervasive for the last 15, 20 years, and what you've seen since then is this sort of like Cambrian explosion of SaaS applications. So there's a different SaaS tool for everything you might want to do in the world, right? It created this huge productivity gain, because I no longer have to write every piece of software. I can just give you some service and I don't have to think about running it either.

So overtime what's happened is that those applications have created – Now, I think the average enterprise has over a thousand different SaaS applications in their organization, and what that creates is this huge amount of surface area. So if you think about a single organization using a thousand different vendors, what you've done is you've taken all these data and you've basically replicated it across all these different environments. So the security posture of your weakest vendor actually becomes a security posture of your data. So if that data is remotely important, it customer records in it, or now things like GDPR, if it just has like an email address in it of an end user, then you as the person who has sent that data to that application vendor, you sent your information off to Grammarly or Notion, like these kind of modern hip SaaS companies, they're now responsible for the data, but you're responsible to your consumer who put that data into your application.

So that situation has caused a fairly untenable environment for a lot of IT admins. They see, "Okay. We're responsible for this data that our customers give us and we're sending it off to all of these other organizations. So we need to make sure that they are being secure about how they handle it." But that type of vendor assurance is very hard to get on a company that's 100 people, or 200 people. Sure, our perspective is you can get the level of security assurance that you need from the big three cloud providers, right? That's why we differentiate between infrastructures as service and software as a service, because we think that enterprises can trust Amazon, Google, Microsoft, to actually manage the hardware and use the services that they provide. But then when they want to deploy that Grammarly tool that's going to key log all the

things that your employees are typing into every text box, instead of sending that data off to the Grammarly servers, why not have that be sent to a private instance that's managed in your VPC and then every time a company like Grammarly would update, you'd get an update notification and you could apply that to your infrastructure.

Grammarly doesn't do this. So you would have to use some other application. A lot of the tools that we actually power are very like specific developer tools. So our platform is used by folks like CircleCI and Hashicorp, NPM, Sysdig, to package up there cloud native applications and then license and distribute it into these large enterprise environments where it can be installed privately into these VPC's.

What that does, that gives the end users control over all of that data. So they don't have to think about adding another vendor who then might – They need to check in to make sure they have PCI compliance and whatever data they're going to put in there. They're never sending that data to the vendor. The vendor is sending the application to the enterprise where they're going to run it in this private instance.

[SPONSOR MESSAGE]

[00:15:45] JM: Triplebyte fast-tracks your path to a great new career. Take the Triplebyte quiz and interview and then skip straight to final interview opportunities with over 450 top tech companies, such as Dropbox, Asana and Reddit. After you're in the Triplebyte system, you stay there, saving you tons of time and energy.

We ran an experiment earlier this year and Software Engineering Daily listeners who have taken the test are three times more likely to be in their top bracket of quiz scores. So take the quiz yourself anytime even just for fun at triplebyte.com/sedaily. It's free for engineers, and as you make it through the process, Triplebyte will even cover the cost of your flights and hotels for final interviews at the hiring companies. That's pretty sweet.

Triplebyte helps engineers identify high-growth opportunities, get a foot in the door and negotiate multiple offers. I recommend checking out triplebyte.com/sedaily, because going through the hiring process is really painful and really time-consuming. So Triplebyte saves you a

lot of time. I'm a big fan of what they're doing over there and they're also doing a lot of research. You can check out the Triplebyte blog. You can check out some of the episodes we've done with Triplebyte founders. It's just a fascinating company and I think they're doing something that's really useful to engineers. So check out Triplebyte. That's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte. Byte as in 8 bits.

Thanks to Triplebyte, and check it out.

[INTERVIEW CONTINUED]

[00:17:35] JM: All right. So you're describing why I love your company, and the example I like of one of your customers is readme.io, which is run by Gregory Koberger who was one of the earlier guests on the show. I really like Greg. He's just a really nice guy, and he's super smart and just kind of hilarious. So he holds a special place in my heart as being an early guest on the show.

But let's say I run readme.io. I am Gregory. Readme.io is a SaaS company that provides readme's. So if you have developer-facing software, whether they are internal developers or external developers, which is a lot of different companies, you need readme's. I don't know you've ever deployed your own ReadMe to whatever, like GitHub pages or whatever, but it's like not a great process. So like I'll give a free plug to readme.io, like if you're developers and you're listening to this, you should at least know that this piece of SaaS exists.

Of course reame.io was a YC company. It was started in the age of selling to SaaS companies, to other SaaS companies, to other modern post-cloud companies, and he got traction. The company got traction, and eventually they got to the point where, well, they are on-prem enterprises. They're ready to buy my really nice ReadMe software.

So when I get to that point, if I'm Gregory, and I've spent years building a sales model and an integration model that targets companies that are on the cloud that are not these "on-prem enterprises", I've got a bunch of software that have already built that is cloud ready. What do I need to do to change my software to be able to deliver it to on-prem customers?

[00:19:29] GM: Yeah, sure. So Greg actually wrote an amazing blog post about this, and I think it's titled like How We Went On-Prem in A Week. So that the real key innovation here is not even that much about replicated. It's about orchestration and scheduling and cloud native applications, right? So as soon as you start to adapt these patterns for cloud native applications, what you're able to do is have a manifest describes all the different components, containers etc., that you can then spin up and install into any environment. Just like you have staging or production, you're not installing that by like manually doing things. You're just running the automation again. So that same manifest can be used to distribute your enterprise customers.

Now often times what happens for companies like ReadMe is that they were leveraging sort of some of these third-party API-hosted services themselves. So maybe they were using something to send emails out. So instead, they need to kind of have like this little workaround in their code that says, “Well, if it’s an enterprise deployment, then make sure that you collect the SMTP information from the IT admin and then we’ll send emails through the SMTP server that’s provided by the enterprise.”

So there’s these little tweaks you have to do along the way that sort of our enterprise integrations. Another thing you might see is that ReadMe, maybe, you have to create an account with your email address and a password, but in a large IT organization, they’d actually would prefer to manage users through a single sign-on service, maybe LDAP or Active Directory. So you’ll have to make some changes to that.

Now we try to offer – We call these like enterprise helper features, things like an LDAP, an Active Directory integration. So instead of you having to build out all that infrastructure, you just plug in to some APIs that we created that are available when you’re deployed on-prem. So we kind of distribute these with your application that make those integrations easier.

So the key would be, for someone like ReadMe, just be containerized. Use Docker, or use Kubernetes. Then from there, abstract away some of these like proprietary systems. If you’re using Red Shift, that's going to be hard, because that's on a portable – It’s on a portable service that lives in AWS. So you might want to shift on to an open source alternative, or if you're using RDS for PostgreS, you might need to use the public containerized version of PostgreS, or maybe you’ll select something like CockroachDB or YugaByte, which both have PostgreS

compliant APIs, but are container kind of cloud native databases. So you'll need to make some decisions around your application architecture to actually leverage and make the entire piece portable.

[00:22:09] JM: Yeah, and so this is what you offer to ReadMe and companies like ReadMe is this –I guess you may call it a framework, or a deployment platform, or a series of steps together with software for getting you and your company to a place where you can offer software to these on-prem enterprise customers. So if you have a product that was built for cloud customers, this is a way to get your software amenable to an on-prem deployment.

[00:22:46] GM: Yeah, exactly. So if you had a multitenant application – Or even the other interesting thing is, what we've seen are companies that have a traditionally on-prem product, maybe they are that JAR product that we talked about earlier, that Java application, and they want to be able to do both SaaS- hosted and on-prem deployments, well the best way to do that today is by using cloud native architecture.

So they'll re-architect their applications in order to offer the SaaS-hosted version, but then once they've done that, there's a company called Jama Software based up in Portland that it is a really early customer an example of this where they had a traditionally on-prem product, have been doing a hosted version for a while, wanted to be able to leverage cognitive architecture, but they knew that if they did that, they would have this massively complex architecture to distribute down to these end customers with all these microservices. So they wanted to mass that complexity, but still leverage it. So that's really where Replicated help them sort of in a similar fashion, but from just a more modern way to do an on-prem deployment.

[00:23:49] JM: Does this is assume that the on-prem company can always consume a Dockerized or Kubernetes-ready version of an application? Aren't there some on-prem infrastructure companies that aren't ready to run Docker containers?

[00:24:10] GM: Yeah. So we really have two different solutions for this. We think about when you're distributing to customers who are kind of cloud native themselves, maybe they have their own Kubernetes cluster that they want to install your application in. We actually just released a really great product to sort of facilitate that, where instead of them deploying the application as a

software appliance, they can actually deploy the application as a fully, like integrate it with their existing deployment pipelines.

So this is the ability to take these manifests and put them into a git repo and then use some type of git ops deployment to get those in your cluster, and then any time the software vendor updates the application, our technology is sitting in the background, it detects that update, it pulls down the latest version, takes whatever custom deployments that the enterprise wanted to apply and then actually merges us together and then makes a pull request into that same git repo with all of the latest updates into it.

So this is one way that you can deploy to these super advanced cognitive companies, but on the other side, as the 95% of the market who maybe isn't ready really familiar with Kubernetes or Docker and they just want sort of an easy to install and manage solution, and that's what the core replicated product has always done, is its sort of been a one line installer, one click updates, the end customer doesn't have to know anything about Docker, anything about Kubernetes. They just sort of run this fast script from their workstation or from the server and then they walk through the setup of this application, the provisioning of additional hosts, everything else, until they have a fully working private instance of that application in their on-prem data center or in their VPC.

[00:25:56] JM: I want to know something about what it was like to build this product, because if I think about – So as you mentioned some of the customers you have that use Replicated. Readme.io is on the simpler side of the examples. I think you also have things like Sysdig, like you mentioned, which is a monitoring software, or I think a logging software or like a security monitoring software I think. You've got Hashicorp, which makes like deployment and distributed systems software.

Were there some engineering difficulties you encountered when you were trying to make a platform that allowed such a diverse range of people to deploy to on-prem environments?

[00:26:40] GM: Yeah. So we've been doing this for four years. So when we first started, Kubernetes was not really been a thing yet. So we had to write our own orchestration and scheduling system around Docker containers. That was – Like many, many companies have

done this. It's a lot of work. It's hard. We focused on building a solution that would really not – It wasn't focused on like this huge scale. It was really more about like how do you manage the complexity of doing like a 5 or 10 node deploy. That was a big challenge.

At the same time, one of the challenges of the replicated platform was when we first launched, like no one was really even familiar with Docker, Kubernetes or any of these concepts. So we were introducing the idea of using containers and orchestrations to a lot of companies and we'd have to help them along that journey in order to get more cloud native ready.

So no since then, we've totally swapped out that original scheduler. Now we rely primarily on Kubernetes as the orchestration and scheduling platform underneath the hood. We also have compatibility with Docker Swarm, because a lot of folks have compose files for their applications. So we wanted to make that an easy onboarding step. But the challenges really come with make this a solution that can deploy lots of different services, and then the actual really complex part – Because, honestly, most of the complexity around deploying these solutions is solved by things Kubernetes or Docker Swarm. The complex part is how do you make sure that the end customer environment meets all the requirements?

So a thing that I think a lot of folks don't realize is that Kubernetes, when it's managed as JKE or AKS, there's a lot of services and connection and things like storage that are happening under the hood that are not just raw Kubernetes, but are actually part of that offering. So we needed to reproduce basically an out-of-the-box distro of Kubernetes that had things like networking solved. So we used Contour, and we used Kubeadm, and we used Rook and [inaudible 00:28:46]. So there's just a lot of additional components you had to deploy other than just raw Kubernetes.

[00:28:53] JM: So we've talked about your company, and we've talked about the problems of on-prem deployments. We've talked a little bit about the engineering. Let's take a step back. Why did you start working on these problems of on-prem deployments?

[00:29:10] GM: Yeah. So my cofounder, Mark, and I, had started a separate company about seven years ago. That company was pretty quickly acquired by a publicly traded SaaS company called Live Person. So they kind of created the idea of live chat support on the web. So we ran

the mobile team there for 2-1/2 years. But while we were there, one we learned a lot about the enterprise software ecosystem sort of how everything works, but the other thing that we saw was there was demand for even a company like Live Person that had been a publicly traded SaaS company for 20 years. There is demand for an on-prem version of the software, and there wasn't really a great solution for distributing that.

Maybe you wanted to package of physical appliance, maybe somebody else thought we could at least two of VM. Mark, my cofounder, he's really the architect and true brains behind Replicated. He recognized that with the introduction of Docker that they would be this new level of application portability that we could leverage to actually distribute these applications into these end user environments. We should realize that that problem was bigger than Live Person. So we left that company and decided to start Replicated as the sort of way to bring this concept to bear for the entire market.

In that time sort of recognized that, “Well, there’s this big opportunity here,” because we’re kind of talking about it before and in kind of referencing like financial services and healthcare and these companies that were born before the cloud as the primary users of like “on-prem software”, but as it turns out, there's actually a lot of very modern companies that prefer these private instances of software.

So if you look at it, a company like Coinbase is actually kind of known for this, where they don't want to use like every SaaS-hosted solution, because the data that Coinbase is holding is worth billions of dollars. So if they are even just giving you the email addresses of their users, that's potentially a security breach. So you see companies like Coinbase or even like Google and Facebook and Airbnb and Uber who would prefer private instances of other people's SaaS software to be deployed into their infrastructure rather than trying to make sure that that vendor meets all their compliance and security requirements.

If you can deploy a private instance of the software and it's the same operational overhead as using the SaaS application, why would you ever send that data off to the SaaS vendor when you can protect it and keep it into your own VPC where that data already lives?

[00:31:46] JM: Coinbase actually uses Replicated, right?

[00:31:49] GM: Yeah. So Coinbase went on the record a couple of years ago talking about how they use several different applications through Replicated. So they'll use our technology to deploy private instances of these different applications into their own – Generally, probably a VPC. I don't have details on where they're deploying those applications, but I wouldn't be surprised if it was a VPC more often than not, because most of the time these end customers are not actually deploying it into a physical data center, but they're using their own AWS VPC, your Google VPC, and then just deploying private instances of the software there. That way it's kind of preventing the application vendor from ever seeing the data.

[00:32:29] JM: VPS, that's acronym for virtual private cloud, right?

[00:32:32] GM: Yeah, exactly. So it's a term that Amazon introduced at first, and then Google has I think – Everyone's kind of started to use it, but the idea is you kind of talked about earlier where on-prem data centers look more like the cloud. Well, VPCs make the cloud look more like on-prem data centers, and they do this by saying, "All the connections and all that happened internally are not going to be exposed to the public internet. They're going to happen over private IP addresses and their encrypted communication on these private networks."

So what that does is it actually facilitates a different model of computing, where the services don't have to be exposed to the public internet. You can actually still use something, like if you've followed up much around the BeyondCorp sort of model of deployment Google has deployed.

[00:33:22] JM: Yeah. We've done a couple of shows on zero trust networking model.

[00:33:26] GM: Sure. So it's funny. When Google launched that BeyondCorp blog post or paper or whatever it was, a lot of people thought that they were saying that Google's only going to use SaaS applications, and it couldn't be further from the truth. What Google was saying is they're going to make sure that there is a publicly exposed access point. So an access proxy for every one of their internal applications.

Now those internal application still all run on private networks, but there is a tunnel between those private networks and that access proxy. So they're not putting their databases on the public internet with these proxy exposed. Those are all on VPCs. The application layers are all on these VPCs. It's just that there's an access proxy that sits in front of these where you can use that to then get authentication into so you can get this sort of course-grained access control from the internet and then get into the VPC from there by using things like a UB key or other username and password-based authentication.

[00:34:26] JM: What's the connection between zero trust networking and VPCs?

[00:34:30] GM: That the – So like that the whole thing about zero trust networking and the way that BeyondCorp described it was that you wanted to be able to get rid of the VPN. So no more VPN. Instead, every application that they use is exposed publicly on the internet, and instead they're basically using – Instead of using VPNs to secure access, they use these access proxies. So Cloudflare has a public version of this that they call Cloudflare Access, which we pair internally at Replicated with another Cloudflare product called Argo. What this does, it takes our Kubernetes cluster where run all these private instances of applications, like analytic servers and BI tools and then exposes access to those only through the internet, but you have to come into this access proxy that Cloudflare hosts for us. So you're accessing all of these services through the "internet", but the services are actually living in a VPC.

So what it does is it enables a company like Replicated or really any organization to not have to use the publicly hosted SaaS multitenant versions to these applications, but instead deploy them privately and then still secure them and make them globally available by using something like the BeyondCorp model.

[00:35:48] JM: Okay. Okay. I think I got it. So Replicated started with this product, the vendor product, allowing people to give on-prem customers a version of their software that could be deployed on a VPC, or in an on-prem environment. That was your first product. Then you moved on to a second product, and the second product is a little bit more subtle, which is Replicated Ship, and I think to understand Replicated Ship, we have to talk a little bit about Kubernetes configuration.

So when we talk about the configuration of a Kubernetes deployment, what are we talking about? Explain what Kubernetes configuration is in more detail.

[00:36:38] GM: Sure. So there's kind of this broad world of Kubernetes configuration. There's a really amazing white paper written by one of the core contributors to Kubernetes, this guy named Brian Grant, and Brian wrote a paper called – I think it's like Declarative Configuration Management for Kubernetes. Yeah, Declarative Application Management in Kubernetes.

What kind of the paper basically does is it gives an overview of all the different options you can use for configuration management. You've got models like templating, and you've got inheritance, and you've got these patches and overlays. Sort of the state-of-the-art in the Kubernetes ecosystem to do this sort of configuration management is to use a product like Helm. What Helm does is it has a lot of different features today and sort of the Helm 2 world. Helm 3 is going to be a little bit more specific.

Helm 2 really basically gives you a way to describe all the different components that you have in your cluster and template out the different things that you might want to have someone, the amount of change per environment. The reason this matters for Replicated is because often times Helm charts are the best way to share amongst the community or from an application vendor to an enterprise that wants to run a private instance of an application in their own Kubernetes cluster. It's a way to share the application and sort of this like package that can then have different values applied, this value.yaml file.

So I think you can think about Helm as a fairly analogous solution to what we've seen in the part or configuration management of that Chef, and Puppet, and Ansible, where you identify the options that you might want to configure and then you can use these values files to actually override those values per deployments. So these kind of like customer supplied values are collected in this values.yaml.

So that's kind of been what most people do for configuration management. Specifically when you're deploying a third-party component that somebody else wrote that Helm chart for, the challenge is that when somebody else writes configuration management, and I you've used Ansible Galaxy, or Chef recipes, you kind of know that, well, the first thing you end up doing is

often times just forking the sort of canonical upstream reference and then making the changes to make sure that it fits into your workflow.

So maybe you're just copying and pasting, not even actually forking it, but you're taking the exposed –The raw sort of chart and then you're making your own custom edits in line and then now you have this fork of that sort of configuration, but it works in your environment. Is that a fairly common pattern you're familiar with?

[00:39:22] GM: I wouldn't say I'm familiar with it, because I might not a programmer, but I mean at this point, like I don't write much code, but it sounds consistent with what I have heard from various interviews.

[00:39:32] GM: Yeah. So this is like a very common workflow. Helm is this great tool. I mean, Helm like came on the scene, it was super needed. It added – It actually filled a lot of the gaps in the Kubernetes like platform overall. It did things like upgrading and rollbacks and just – It's a really powerful tool, but overtime, the Kubernetes API has gotten a lot more powerful and it's kind of like there's some overlap now between that functionality. Then you've also just started to see where, again, everyone is copying and pasting these Helm charts and making their own version.

So the biggest challenge that creates is that when you take a forked Helm chart and now the upstream updates, you're basically trying to manually reconcile the difference between the version that you created and the new version that came from upstream, and that's this like complex manual toil type task, where you're like, "Okay, I got to figure out what changed. What do I do to mine and what what's updated in the new one."

So people often don't even do it, they just like – Maybe they'll update the images. Maybe they'll do something else, but this becomes a thing that – Like the cluster operator ends up being responsible for this entire forked Helm chart. So we never thought that was a great experience. Then Brian's paper beyond just actually describing all of the sort of existing solutions for configuration management, actually lays out a fairly comprehensive overview of a new way to do configuration management that's declarative. It sort of comes out that he wants this system of patches and overlays to work.

So we really loved this paper, and we started working some of the concepts into what we're building at Replicated, and the Google open-sourced a project called Kustomize, with a K, that actually implements almost this entire paper, right? So what Kustomize did that's so important, so different, is that it said, "Look, you should be able to take an upstream configuration," and then instead of needing to like copy and paste it and then make edits to it directly, you should be able to edit it or like customize it by describing a patch, and a patch is like a little snippet of YAML that together with this tool can actually be used to traverse the application Kubernetes manifest and then make changes inside the manifest.

So it's sort of done as like a – You're basically separating out the configuration changes that you might want to make to that application manifest and writing it separately. The beauty of this system is that by separating those things out, any time the upstream actually updates, you never changed it. So you can just update it directly and then your patches can still traverse the new version of that updated manifest and apply the changes that you want to have.

[SPONSOR MESSAGE]

[00:42:35] JM: MongoDB is the most popular, non-relational database and it is very easy to use. Whether you work at a startup or a Fortune 100 company, chances are that some team or someone within your company is using MongoDB for work and personal projects.

Now with MongoDB Stitch, you can build secure and extend your MongoDB applications easily and reliably. MongoDB Stitch is a serverless platform from MongoDB. It allows you to build rich interactions with your database.

Use Stitch triggers to react to database changes and authentication events in real-time. Automatically sync data between documents held in MongoDB mobile or in the database backend. Use Stitch functions to run functions in the cloud.

To try it out yourself today, experiment with $10 in free credit towards MongoDB's cloud products; MongoDB Atlas, the hosted MongoDB database service, and Stitch. You can get these $10 in free credits by going to mongodb.com/sedaily. Try out the MongoDB platform by

going to mongodb.com/sedaily. Get $10 in free credit. It's the perfect amount to get going with that side project that you've been meaning to build.

You can try out serverless with MongoDB. Serverless is an emergent pattern. It's something that you want to get acquainted with if you're a developer, and getting that $10 in free credit to have a serverless platform right next to your Mongo database is really a great place to start. So go to mongodb.com/sedaily. Claim that credit, and thanks to MongoDB for being a sponsor of Software Engineering Daily. I love Mongo DB. I use it in most of my projects. It's just the database that I want to use.

Thanks to MongoDB.

[INTERVIEW CONTINUED]

[00:44:43] JM: To get to the core of what you're saying is there the way that many people have ended up in their Kubernetes deployments and their Kubernetes git management, is that they are hitting merge conflicts or branch conflicts with the way that they're updating their configuration files, because their configuration files will end up diverging from the project that they are based on, some open source project that they're based on. Then when the open source project has an update, they get a merge conflict.

[00:45:20] GM: Yeah, exactly. The reason you get that merge conflict is because it's basically impossible for a Helm chart to describe all the different like little things that a cluster operator might want to configure about that chart.

[00:45:35] JM: Helm chart, by the way, this is a description of a distributed systems deployment, like basically a distributed application that would go on to your Kubernetes clustered.

[00:45:47] GM: Yeah, exactly. It's a way to like kind of – They call it the package manager for Kubernetes, which is really powerful concept. If you can package up distributed applications into a single sort of artifact or something that you can then use to deploy it, that's really amazing. But the challenge with this is that as the chart maintainer, I can only like make configurable so many

different options. So then as someone who wants to operate that application, if I want to make any changes, if I want to use other parts of the Kubernetes API that are not templated in that original Helm chart or if I want to add in and integrate a CRD, which are these custom resource definitions that are really popular way to extend the Kubernetes API, then what I have to do is basically copy and paste or fork that original Helm chart, make all my updates and then now I'm stuck managing that entire Helm chart, right? That's been just how the world has worked for the last however many years, but it doesn't have to be and it's not like that anymore with this new tool that Google introduced, this new sort of methodology that they introduced with customize, and it's actually being integrated fully into Kubernetes. So you'll be able to Kubectl apply any folder with a customization.yaml in it and it will just know how to natively traverse the application manifest with these updates that you want to make.

[00:47:10] JM: I just want to reinforce this a little bit more, because this is like people are – I'm sure there are some listeners right now who are like, "What the heck is – I'm so lost right now. This is some –" Even people who are working with Kubernetes.

So like let's take an example, like Linkerd. So if I am deploying a service mesh like Linkerd on to my Kubernetes cluster, I might want to customize it. I might have some custom configuration changes that I want to make to Linkerd, and then later on Linkerd is going to be updated and I'm going to have some issues there. Can you just explain why this different form of configuration management will be useful for a project like Linkerd.

[00:47:51] JM: Yeah, sure. So basically our perspective is that Helm charts – So the Linkerd Helm chart should describe sort of the basic way to get Linkerd up and running in your cluster or in the sort of canonical way. However, what you see overtime is that these Helm charts end up expanding. So if you look at Prometheus, I think the first Prometheus chart, the values YAML is maybe 415 lines. Today it's over a thousand lines. The reason for that is because as the community has started to use it, more and more custom configurations – So you want to access different parts of the Kubernetes API. What you have to do, like say I want to use pod security context, which is sort of not super common, but like maybe 10% of end users might want to use that as a thing they define.

Well, if that's not in the original chart, then first thing I'm going to do is fork it. Add pod security context into my deployment and then I'm going to manage that and maybe I'm going to make a pull request into the upstream with that change into it. So what Prometheus has done has except that a lot of these make this configurable type pull requests. So they're constantly managing all these pull request to make some new attribute configurable, and they're doing that because basically it would – As it turns out, that like basic deployment of a project is not what people end up using for their production deployment. They end up having – Maybe that's what 90% of it is, but they end up adding their own custom 10%, because they do things a little bit differently.

So as everybody adds their own custom 10%, you end up with this huge and complex Helm chart, and the way to avoid that is say, "Well, instead of accepting all these upstream pull requests to extend and expose template versions of all these different components, we should have everyone add in their own customizations themselves and manage those customizations externally."

So that's what customize tries to introduce. Then Replicated Ship, which is the open source project that we created really operationalizes that. So it says that it's going to help you create one of these customization files by pulling in a Helm chart, templating out all the YAML, giving you deployable Kubernetes manifests that you can then click into and then make little updates and we're going to like pre-calculate all the patches for you and them are to save that in a way that you can deploy it to your cluster straightaway. If you want, we'll actually watch the upstream repo, and anytime the upstream updates – So let's say the upstream Prometheus chart updates, we're going to download those updates, we're going to take your patches, merge those on top of the updated version and then make a pull request into your private git repo with all of the updated manifest and your custom configuration still preserved. So you're going to get these automated pull requests every time you update the application, or the application is updated from upstream.

[00:50:51] JM: This is not something that's only useful for on-prem companies, right? This is like for anybody who is using a lot of different open source projects within their Kubernetes cluster.

[00:51:06] GM: Yeah, exactly. So this is a tool that we created basically for any Kubernetes cluster operator. Ultimately, our vision of the world is that we will all run private instances of applications, may be hundreds or thousands of different private instances of applications, and in order to run those, we need a system like Kubernetes to actually manage the operational overhead and to make sure this is all automated.

So in order to have a world where there are thousands of private instances both open-source and commercially supplied applications, well, we need to operationalize this better, and we need operationalize this in a way that I can add these custom configurations without needing to fork everything and take over all these operational management.

So Ship is designed to really focus and reduce the amount of YAML that a cluster operator is managing, because if you fork the Prometheus Helm chart, you're taking over tens of thousands of lines of YAML, and if you just apply a patch, maybe you're going to manage 100 lines of YAML. So the amount of like actual YAML under management for a cluster operator who's forking Helm charts versus one who's using Replicated Ship plus customize is going to be orders of magnitude different.

[00:52:25] JM: So you mentioned a few interesting things there. I want to understand your vision for what your company does, because you've got two products right now. If I squint, I can see some and similarities between them, some overlap between them. The first one is, like we said, this product for allowing SaaS companies to have on-prem software. The second one is this configuration management tool. What is the business? Like what's the long term business vision? What's your vision for what Replicated becomes?

[00:53:00] GM: Yeah. So these two are actually very tightly linked, and the reason they're tightly linked is if you can imagine the current vendor solution, like you need a way for proprietary applications to package up and distribute into the same kind of workflow. So if we think about Ship is the future Replicated and cluster operators are using it to automatically manage the third-party components that they're deploying into their cluster, those third-party components can be these open source solutions like Prometheus, but they could also be a private instance of Gradle.

So Gradle is actually an application vendor that uses Replicated and then distributes their applications through the Replicated Ship platform. So their end customers can run Gradle enterprise in an existing Kubernetes cluster next to a lot of other different applications that are deployed in that cluster, and the way that they distribute that license and keep it updated and have all these release notes come through is by using the replicated vendor tooling in order to distribute it down to them.

[00:54:04] JM: I see. Let's start to wrap up and talk more about the broad world of, I guess, how infrastructure is being reshaped by Kubernetes and the cloud. These are like these two forces that are so strong. They're having – And obviously Kubernetes and the cloud are related to one another, but they're really reshaping how software is procured. They're changing the power dynamics of different companies.

Is it hard to be a vendor in this space? There's a lot of noise, but there's also a lot of money being spent. What has been your experience building a business in the midst of all these change?

[00:54:49] GM: Yeah. I mean, so change is great for startups, because what happens is all the traditional vendors kind of get shaken up, and it creates his new opportunities for a startup to come in, find a solution to problems that haven't been solved in the past, like the idea of automated pull requests for every time your third-party applications update is only possible because we have a Kubernetes API that's well-defined and understood.

That it that makes possible the idea of this declarative configuration management that customize provides. So there're all these changes that are happening in the market, and if you're paying close enough attention, you can actually see the ones that are really creating these like platform shifts, right? These platform shifts are what new big businesses are built on. So we look for those all the time. The reason that we're constantly paying attention to what's going on in the Kubernetes ecosystem is because it's moving so fast and it's creating so many different opportunities. It's really shaking up everything that's happening in infrastructure.

We think the next thing it's really a shakeup is the SaaS ecosystem, right? I mean, the funny thing is that a lot of people, a lot of investors, really believe that SaaS has a bright future

because the amount of software spend on on-prem software is still enormous. I think the SaaS ecosystem is maybe 100 billion a year and the on-prem ecosystem is about 350 billion a year. So most investors would tell you that those worlds, it will become SaaS everything at some point.

Replicated, what we see, it's probably more shades of gray than it is black-and-white. So we see a world where folks like Coinbase or large financial institutions or governments are really just any company that knows that their data that they have is important. We see a world where they can run these private instances of applications in a Kubernetes cluster secured through BeyondCorp model, done in a way that it doesn't take any additional operational know-how from their IT team to manage a thousand different applications that are deployed into that cluster, and it's just automated in a way that it's running. Maybe there's a there's operators and CRD's that are acting behind the scenes to help keep databases going in a more self-healing way.

So we see this sort of new frontier of enterprise software and we know that the demand is there from the end-users. Like there's a huge amount of spend that's happening in this space, and we know that not many other folks are really focused on what does it take to enable application vendors to make this transition and to go from maybe a traditionally SaaS model to be able to offer these private instances for enterprises to deploy in these sort of more complex environments. So that's what Replicated really focuses on and that's what we're trying to help bring to market.

[00:58:07] JM: All right. Well, Grant, it's been really fun talking to you, and I think you got a bright future in your business. It's a really interesting company. I think this space, it's so big, I've seen people try to size the market of life cloud natives or cloud software or Kubernetes market. I don't think it's like sizable at this point. I think it's going to grow and grow and grow and grow. It's so big. There's so much opportunity.

[00:58:36] GM: It's true. It's funny, I think you get that asked the question a lot, "What's the size of the market's?" It's like, "Well, I mean, the economy – I don't know. What's the answer?" If software east the world and Kubernetes easts software, you need automation.

It's funny. I always tell this story around why Kubernetes matters so much, because they've referenced the Twitter Fail Whale. Did you remember when Twitter, they used to see the Fail Whale all the time?

[00:59:03] JM: I do. Of course.

[00:59:04] GM: Yeah, and like that was – What? 10 years ago, right? That was this big, successful Silicon Valley company and they couldn't figure out how to deploy reliable software. I mean, hell! Salesforce still puts out like service interruption updates on their website that they're going to take their site down. Like 10 years ago, we didn't know how to build reliable software. Really, no one did. We didn't have the primitives. I say no one, but one company knew how to build reliable software, and it's why if your internet feels flaky, you go when you ping google.com to see if it's up, right? See if your internet is working. You trust that Google is up more likely than you trust your internet is up.

So what happened was Google took this system, they wrote all these software in order to make their services reliable, and when they did that, they called it a Borg, and it was basically container orchestration and then they open-sourced it as Kubernetes and they gave this primitive to the world that now allows anyone to create truly reliable software. I mean, they wrote the book on it, the SRE book, site reliability engineering.

With this primitive – So kind of circling back to Twitter, they took the concepts that Google was using and they built their own version of it. The called it Mesor and they kind of like – Or they took the Mesos project and they really used that internally. So they used a solution that was container orchestration and scheduling in order to solve their fellow problems, but now more and more companies have access to build truly reliable software, and that's going to just change how we think about and how we use software. So the opportunity is just enormous.

[01:00:47] JM: Okay, Grant. Well, great talking to you. Thanks for coming on the show.

[01:00:50] GM: Yeah. Jeff, thanks so much. Have a good one.

[END OF INTERVIEW]

[01:00:55] JM: This podcast is brought to you by wix.com. Build your website quickly with Wix. Wix code unites design features with advanced code capabilities, so you can build data-driven websites and professional web apps very quickly. You can store and manage unlimited data, you can create hundreds of dynamic pages, you can add repeating layouts, make custom forms, call external APIs and take full control of your sites functionality using Wix Code APIs and your own JavaScript. You don't need HTML or CSS.

With Wix codes, built-in database and IDE, you've got one click deployment that instantly updates all the content on your site and everything is SEO friendly. What about security and hosting and maintenance? Wix has you covered, so you can spend more time focusing on yourself and your clients.

If you're not a developer, it's not a problem. There's plenty that you can do without writing a lot of code, although of course if you are a developer, then you can do much more. You can explore all the resources on the Wix Code's site to learn more about web development wherever you are in your developer career. You can discover video tutorials, articles, code snippets, API references and a lively forum where you can get advanced tips from Wix Code experts.

Check it out for yourself at wicks.com/sed. That's wix.com/sed. You can get 10% off your premium plan while developing a website quickly for the web. To get that 10% off the premium plan and support Software Engineering Daily, go to wix.com/sed and see what you can do with Wix Code today.

[END]