


SECURITY BOULEVARD

Home ▾ Security Bloggers Network ▾

Webinars ▾ Chats ▾ Library

Home » Security Bloggers Network » 20 Developers and Kubernetes Experts Reveal the Biggest Mistakes People Make During the Transition to Kubernetes

20 Developers and Kubernetes Experts Reveal the Biggest Mistakes People Make During the Transition to Kubernetes

 by Mike Broberg on December 20, 2018



Making the transition from virtual machines to containers is a complex process that can take some time, particularly for larger, more complex environments. Users are drawn to Kubernetes' container-centric environment, as well as its ability to enable portability across infrastructure providers. Kubernetes also offers broad applicability; for the most part, an application that runs well in a container will run well on Kubernetes. These, along with myriad other benefits, are what make the transition to Kubernetes worthwhile for many applications. Not up-to-date on the ins and outs of Kubernetes? Check out our list of [50 Useful Kubernetes Tutorials for IT Professionals](#) to get started.

Because the process can be both lengthy and complex, mistakes are common during a transition. First, it's important to understand that [Kubernetes](#) is not a silver bullet. Organizations that adopt container orchestration platforms like Kubernetes before they really understand the technology are more vulnerable to configuration errors. There are also some important [Kubernetes security](#) considerations, such as blast radius (how far a malicious party can gain access beyond the initial point of compromise), that leave certain components of a cluster more vulnerable. That's why it's important to build security into your deployment as early as possible. To find out where your security

If you're ready to get started with your infrastructure transformation, there are other pitfalls you'll want to avoid. To help you get off on the right foot and avoid common mistakes, we reached out to a panel of developers and Kubernetes experts and asked them to answer this question:

“What’s the biggest mistake people make during the transition to Kubernetes?”

Meet Our Panel of Developers & Kubernetes Experts:

- | | | |
|---|--|--|
| <ul style="list-style-type: none">▪ Andy Kennedy▪ Kris Nova▪ Alexandr Kosarev▪ Skyler Layne▪ Radhesh B. Menon▪ Cris Daniluk▪ Ian McClarty | <ul style="list-style-type: none">▪ Demi Ben-Ari▪ Dan Garfield▪ Mark Stadtmueller▪ Tom Petrocelli▪ Brent Stackhouse▪ David Greenstein▪ Jon Friesen | <ul style="list-style-type: none">▪ Davy Hua▪ Igor Drobiazko▪ Eric Bailey▪ Chai Bhat▪ Ben Bromhead▪ Oleg Atamanenko |
|---|--|--|

Read on to find out what mistakes you could be making when transitioning to Kubernetes (and how to avoid them).

Note: There is no implied ranking, preference, or endorsement in the content that follows. The views and opinions expressed in this article are those of the authors and do not necessarily reflect the policies or position of Threat Stack, Inc.

Andy Kennedy

@tier2consulting



Andy Kennedy is the Managing Director of Tier 2 Consulting, the premier Red Hat Middleware Partner for the UK and Ireland. Tier 2 are experts in creating custom software solutions for web, mobile, and enterprise applications.

“Very often making open source software such as Kubernetes ‘usable’ in a corporate environment will require...”

For example, to deliver a rich set of orchestrated services, Kubernetes relies on other

services provided by open source projects – such as registry, security, telemetry, networking, and automation. Organizations need to recognize this, and factor that into their implementation plans – or use an ‘enterprise ready’ product such as Red Hat’s OpenShift Container Platform which already has these services integrated and tested.

Kris Nova

@heptio



Kris Nova is a Senior Developer Advocate at Heptio. Kris is a Kubernetes maintainer, and a maintainer of Kubernetes kops. She also created Kubicorn. An active participant in the community, she dreams that one day she can help make Kubernetes easy to install and manage on any cloud platform. She’s looking forward to defining and solving day 3 management concerns for Kubernetes.

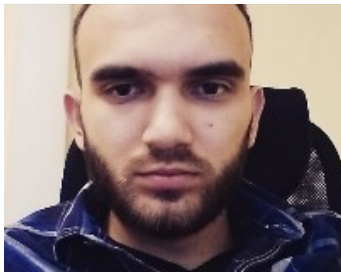
“Moving a large, legacy (monolith) application is different from moving a smaller application already structured for orchestration...”

The glaring risk is a need to audit the systems running the monoliths. Even discovering what dependencies are needed on the host system is a large amount of effort for engineering teams. After ensuring that all dependencies are met and being tracked somewhere, a user could then begin to audit the application itself to ensure that it would behave as expected in Kubernetes.

Be aware that a monolith application can mostly work and not exit despite there being data loss or corruption. Kubernetes would ensure that the pod is healthy, but may take no action on a pod desperately needing to be restarted. In short, migrating a monolith requires additional preparation and thought to avoid these types of issues.

Alexandr Kosarev

@BelitsoftCom



Alexandr Kosarev is a department manager at Belitsoft. He has been working in IT for over six years, starting as a web developer, then moving to management positions. Alexandr specializes in mobile and web applications for eLearning, Financial, and several other domains.

“People can make lots of mistakes when transitioning to Kubernetes. I would focus on the three most widespread...”

1. Buying too much server power: For one project, we used a rather average configuration. CPU load was around 5%, and memory use was around 10%. And we still have the option of autoscaling the app. So don't waste your money on power you don't need.
2. Using a session persistent load balancer: This creates a bottleneck that makes Kubernetes autoscaling almost useless. The right approach is to adjust your app so it works well with the balancer, not the other way around.
3. Persistent storage for temporary data: Most applications need temporary files (like thumbnails). And of course, you'd like to share them between the instances to reduce the load. You can buy additional persistent storage space for your nodes.

Skyler Layne

@Indellient



Skyler Layne is a Senior DevOps Specialist at Indellient where he helps organizations reach their IT-related and business goals. Skyler is skilled in Java, Ruby, and Python, and has worked on numerous development projects since earning his Bachelor of Software Engineering degree from Toronto's York University. Skyler's hobbies include playing with his cat and collecting antique pens.

“The biggest mistakes people make during the transition to Kubernetes are...”

1. Boiling the Ocean: People feel the pull of Kubernetes and all the benefits that it can have. Generally, those benefits involve huge overall architecture changes to your application. In this case, it's suggested to move forward like smoking meats — low and slow. It's difficult enough getting your app to run in containers, so start with the low hanging fruit

2. Ephemeral Nature: Another pain point in Kubernetes is the ephemeral nature of containers. In Kubernetes, and containers in general, your infrastructure is treated as cattle instead of pets; it's going to go away as it should. This is an important point and should be a major principle when you start to bring applications into containers. Containers are not VMs, nor should they be treated as such. Stateful data should be handled in a responsible way.
3. Security: Often people overlook the fact that yes, your Kubernetes cluster nodes are actually running on VMs. While this might not be important to someone who is consuming a Kubernetes cluster, it's important to keep in mind that any compliance regulations governing your application stack also fall onto the Kubernetes Node VMs.
4. Overuse of Resources: When you have a ton of resources in your hands and a problem to solve, and you want to make your application run faster, organizations still tend to throw money at it and request more resources than they need. This might leave resources idle if you haven't planned ahead (minimum needed vs. maximum allowed). You can overcome this as follows:
 - Benchmark your application and optimize it for constant use of maximum resources needed.
 - Define request and resource limits so Kubernetes can more effectively auto scale your pods and containers.
 - Use [horizontal pod auto scalers](#) when applicable.
5. Monitoring and Resource Exhaustion: Overuse of resources may rise from a lack of monitoring. Implementing an effective resource monitoring system usually takes time because it tends to get deprioritized by the everyday tasks faced by DevOps teams. Monitoring systems are critical to good Kubernetes usage by the people deploying to it. The lack of monitoring may lead to resource exhaustion. As app developers climb the learning curve, organizations end up with a massive cluster of reserved idle resources. Best practices are key (resource requests vs. limits) such as:
 - Promoting workshops on the correct utilization of the cluster. Teach developers how to extract the best value from it as they climb the learning curve.
 - Prioritize the creation of a monitoring system so teams have a clear view of their utilization of the cluster.

Radhesh B. Menon

@RobinSystems



Radhesh B. Menon joined Robin in 2018 with over two decades' of experience driving market success for leading enterprise software products and technologies. Previously, he was General Manager for OpenStack at Red Hat, responsible for driving business, product management, partner, and worldwide go-to-market strategies. As Global Leader for OpenStack, he helped bootstrap the business and establish the Red Hat OpenStack Platform as the leading open private cloud solution across enterprise and Telco/NFV segments. Prior to Red Hat, he held various product management and product marketing roles at Microsoft across Azure, System Center, Hyper-V, Windows

Institute of Technology, India and an MBA from Simon School, University of Rochester.

“By far the biggest mistake customers could be making is to...”

Ignore Stateful and Data intensive applications when modernizing applications using containers/Kubernetes.

Containers and Kubernetes have become quite the rage given the focus on digital transformation, DevOps agility, and cloud-native technologies. Since containers have inherently been stateless (i.e., don't have persistence), the power of Kubernetes has been limited to running the full stack of the stateless applications across the complete development, deployment, and production lifecycle.

A recent survey conducted by CIO.com revealed that 68% of IT Managers responded that significant improvements are needed to run stateful workloads on Kubernetes. Automated application deployment, performance isolation, and managing environment/application data consistency in production, dev, and test were cited as the top three challenges when it comes to running stateful applications according to the survey respondents.

A complete solution for deploying and managing a stateful application using Kubernetes involves integrating a few critical elements: 1) highly performant storage, 2) flexible network stack with persistent IPs, and 3) an application management layer to abstract complexity and enable self service. The Robin Hyper-Converged Kubernetes Platform is precisely that: a purpose-built solution for extending Kubernetes to stateful workloads so you can deliver on business needs faster, reduce costs, and future proof your enterprise.

Cris Daniluk

@CrisDaniluk

@RhythmicTech



Cris Daniluk leads [Rhythmic Technologies](#), an innovative, compliance-oriented managed cloud and security services firm based in the Washington, D.C. area. Before founding Rhythmic, Cris was responsible for project management and business development at Claraview, where his work in securing projects worth over \$100 million helped key the company's acquisition by Teradata.

The complexity in running highly available, secure applications on top of Kubernetes. It is incredibly easy to get a Kubernetes cluster up and an application running in it, but “up” and “production ready” are very different states. Service health checks, monitoring, deployment strategies, networking, and container security all need to be thought through, and there are not well-established best practices and patterns to lean on yet.

Modern infrastructure is very resilient, and Kubernetes is unlikely to improve availability and performance unless the application is built to leverage its strengths. Specifically, Kubernetes is best for applications committed to a loosely coupled microservices architecture. And, it is best adopted by teams who understand that Kubernetes is not a shortcut but rather an opportunity to improve application performance and availability once properly understood.

Ian McClarty

@phoenixNAP



Ian McClarty holds an MBA from Thunderbird School of Global Management. He has over 20 years' of executive management experience in the cybersecurity and data center industry. Currently, he is the CEO and President of [PhoenixNAP Global IT Solutions](#), which employs a staff of over 600.

“The biggest mistake you can make when transitioning to Kubernetes is...”

To go the DIY route without having the right level of expertise in house.

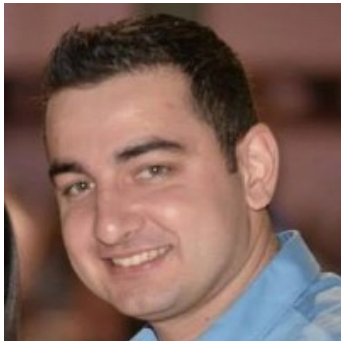
Kubernetes is not a single executable running on a single machine. It is, instead, an ecosystem of different applications and network layers that are closely integrated to produce the final solution (Docker, etcd, SDN overlay, Ingress Controllers, LB, kubelet, kube-proxy, kube-apiserver, etc.). It requires a certain level of expertise to get it running, let alone maintain in production with your software running on top.

Fortunately, emerging as the leading technology of its kind, it has become ubiquitous, and it is offered by all major PaaS providers. It also has integrations with all major technologies, even those that are theoretically direct competitors. Hence, if you intend to get rolling with Kubernetes without too much hassle, don't reinvent the wheel.

ing Kubernetes easy, for Kubernetes creates a highly elastic and dynamic environment where pods are shifted according to load. You need to make sure your application does not make any assumption about the node they run in and that they behave impeccably on startup and shutdown since these operations will become much more frequent than usual.

Demi Ben-Ari

@panorays



Demi Ben-Ari is the VP R&D and Co-Founder of Panorays. Demi brings to Panorays his expertise in building communities and networks – both online and offline. His technical background enables him to provide visibility into companies' seeming blind spots and build large systems to empower users through insight-sharing. Demi is a recognized Google Developers Expert, co-founder of Big Things – one of the largest Big Data communities – and of the local Google Developer Group Cloud. He is a renowned international speaker, presenting Big Data solutions and distributed and scalable systems to both technical and management teams. Prior to co-founding Panorays, Demi was responsible for Big Data technologies at Windward and before that, he spent eight years in the Israeli Air Force as a Senior System Architect for missile defense systems. Demi holds a BSc in Computer Sciences and Management.

“One of the biggest mistakes people make when deciding to transition to Kubernetes and, in general, a container-based solution, comes from...”

The problem that I like to call Dismantling the Monolith. This is thinking that every functionality they have in their current system (Monolith) can be broken down to microservices and just packed into containers, and then you'll have a scalable and decoupled environment with ease of deployment like all the shiny promises that Kubernetes gives us. Instead, what you end up with is what I call a Microlith, a bunch of services that are highly coupled and that you can only deploy together, without solving the initial problem of bulky and risky deployments.

When coming to a process of migration from your Monolith to actual microservices, you'll first need to separate your data accordingly to have really decoupled services and then to integrate with the best practices that the opinionated orchestration framework of Kubernetes provides us. Only then can we leverage the years of experience that Kubernetes brings to the table.

...leverage all the great features of Kubernetes, and it's okay!

Provide a solution that will solve your current problem and iterate on it, and then, when you get the hang of it, you can move to more complex tools like Helm, the Kubernetes Package Manager, for deployment.

Dan Garfield

@codefresh



Dan Garfield is the Chief Technology Evangelist at [Codefresh](#).

“Oh boy, I could talk for a long time about mistakes people make when first adopting Kubernetes. The biggest one I’ve seen is...”

People thinking they’re going to build their own clusters from scratch. I’ve heard this dozens of times from engineers, and I always tell them they’ll regret it in six months. Sure enough, they all end up going with managed solutions, whether that’s cloud-based in the form of clusters on Azure, Google Cloud, Amazon (or many others), or on-prem like Rancher, OpenShift, or GKE on-prem.

The second thing I’d mention – and it’s akin to the first – is that many people decide that Kubernetes and containers are just a deployment change. While it’s possible to do that, you won’t be getting the full value of Kubernetes if you treat it as the last mile. Instead, the entire code development workflow should revolve around getting containers onto Kubernetes. For example, in Codefresh, we’ve reimaged the entire engineering and DevOps workflows just to streamline for delivering containers to production. This saves us tons of time on debugging, testing, release management, you name it. Remember: The number one reason people choose Kubernetes is that they want to increase their development velocity, and they can’t do that if they just treat Kubernetes like it’s a destination.

Mark Stadtmueller

@LucdDIS



Mark Stadtmueller is VP of Product Strategy at Lucd where he is responsible for driving requirements for Lucd's enterprise AI end-to-end Platform as well as leveraging Lucd AI innovation and intellectual property to deliver the digital transformation that Lucd customers need.

"To me, the biggest challenges are..."

Deploying Kubernetes in a way that is independent of and allows flexibility for various deployment infrastructures including public cloud, private cloud, and on premises. Most companies need flexibility in deployment infrastructure choice, and so when deploying Kubernetes, maintaining that flexibility is the biggest challenge.

Tom Petrocelli

@AmalgamInsights



Tom Petrocelli is the Research Fellow for DevOps and Collaboration at Amalgam Insights. His area of interest is collaboration and new ways of work, developer tools, IT project efficiency, governance, and methodologies, as well as DevOps. Most recently, Tom worked for a large, global, banking corporation. He is an experienced marketing, technology, and business executive with 30+ years in the computer technology industry. Tom's background spans software engineering, systems architecture, IT, product management and marketing, and general management.

"From our vantage as outside observers, it's trying to..."

Apply Kubernetes to all applications. It works best in microservices architectures where lots of small services, services that can be placed in containers, are the norm. Trying to re-architect applications to fit Kubernetes is not the point. The point is to better manage container clusters which are not right for all applications.

Brent Stackhouse



Brent Stackhouse, CISSP, CISM, CRISC, GWAPT, CCSK, a risk management professional with deep technical experience, leads the WP Engine Security and GRC teams in their ongoing efforts to protect WP Engine and its customers. As a 20 year security veteran with wide-ranging experience that includes providing information security services to financial institutions and global tech companies, Brent knows what works in securing cutting-edge, cloud-scale environments.

“A common mistake when deploying Kubernetes is...”

Not properly protecting the cloud instance metadata API, as nodes in a Kubernetes cluster typically have powerful instance roles, and the tokens are accessible to workloads by default.

David Greenstein

@coda_global



As a Business Solutions Architect with Coda Global, a cloud-native consultancy and application developer, David Greenstein helps clients align their technology tools and business strategies to meet their current business needs while future-proofing application projects to scale and innovate as the client's business grows.

“Kubernetes is a technology that brings elegance and order to one of the most complicated yet powerful and transformative technologies out there: Docker containers...”

It fills in many of the gaps found when trying to run containers on just the Docker engine alone such as networking, persistent storage, and exposing containerized services to clients. Kubernetes brings a maturity to this technology, enabling customers to achieve greater scale and simplified container orchestration compared to running just Docker alone.

Application packages developed in the wild are not meant to run in containers. Containers running in the wild are fit for production use or to inherently implement a cloud-native application architecture, which takes advantage of more strengths and value propositions of Kubernetes than running an entire application stack in a single container. It is common, in my experience, to find that organizations looking to adopt Kubernetes mistakenly try to leverage images found in the wild as-is, without scrutiny or having a deeper understanding of the architectural patterns which are best suited for running in containers. Such anti-patterns, or suboptimal application architectures, reduce the value of Kubernetes and minimize the number of features the organization can capitalize on.

Anti-patterns come in different shapes and sizes, but the more common ones we see stem from organizations not knowing the strengths and weaknesses of containers.

This leads to misalignment of technology and business objectives often manifested as treating containers like virtual machines, for example, instead of the ephemeral components of an application they are meant to be. Containerized microservices are the cornerstone of cloud-native applications, and as such, they do not necessarily come automatically from deploying Kubernetes and selecting a Docker image and/or Helm chart from an official repository. The structure of the application and implementation details can show you that the image may have been intended to ease development blockers, but organizations will assume they can take that image to production and find unintended consequences the hard way.

If there is one piece of advice I would give organizations looking to adopt Kubernetes, it would be to separate their greenfield opportunities from other workloads, and to architect the best possible technical solution for the requirements based on fit for use and best practices. This will help identify what an anti-pattern is, avoiding the pitfalls others have made, to ensure that the organization achieves the acceleration and value of Kubernetes without having to go to the school of hard knocks.

Jon Friesen

@jonfriesen



Jon Friesen is an engaged and enthusiastic developer creating highly available scalable software, working with microservice systems handling tens of thousands of active daily users in multiple cloud landscapes. Inspired by a fascination for containers, Kubernetes,

“With all the excitement around Kubernetes and the abstractions that make deploying, managing, and scaling high availability cloud applications easy...”

The most impactful mistake during the transition is using namespaces in lieu of separate clusters for pre-production deployments.

Namespaces are virtual clusters that run on the same hardware and Kubernetes infrastructure. The number of resources that are shared puts the production deployment at jeopardy in several regularly occurring situations.

At the lowest level, the hardware is shared, each namespace consuming what it needs. If there is a problem with your pre-production deployments, they may take resources from a production deployment, degrading the customer experience.

One level up is the Kubernetes infrastructure that handles and manages all of these virtual clusters. Testing a new Kubernetes version with breaking changes will bring your production environment down. This removes core value when looking for issues before updates land in the customer-facing systems.

The solution to all this is to use separate physical clusters with their own Kubernetes versions and resource allocations. Separating these concerns produces environments that can be reliably tested without fear of affecting production systems used by customers.

Davy Hua

@ShiftLeftInc



Davy Hua is a leader and entrepreneur who is on a lifelong mission to optimize complex infrastructures. During the last two decades, he has specialized in designing and managing complex DevOps infrastructures. After working as an Engineering Manager, Principal Engineer, and Architect as well as founding his own company, he is presently the Head of DevOps at ShiftLeft Inc, an application security startup.

“The single most disruptive mistake people make during the transition to Kubernetes has to do with...”

Overlooking the proper setting of resource constraints of all the running containers. The possibility of unexpected and intermittent service disruption, crashes, and/or latency

Predict 2019 Virtual Summit

Featured Blog »

FORTIN

Fortinet All Blogs

Connecting Passengers to their P
Cruise Experience

Fortinet All Blogs

IT Leaders Are Concerned About
Security

Fortinet All Blogs

Proactively Addressing New Cybe
Trends in Healthcare

Subscribe to our Newsletters

Get breaking news, free eBooks a
upcoming events delivered to you

Your Email

View Security Boulevard [Privacy Policy](#)

Subscribe Now

Most Read on the Boulevard

Best of 2018: 4 Things to Know About th
Protocol

Walmart Moves Closer to ‘Minority Repc

5 Ways Hackers Killed the Sandbox — a
Do About It

Without setting CPU and memory constraints for each of the running containers, certain containers may become resource hogs and leave little for their neighbors. This will essentially create a noisy neighbor problem.

This problem further amplifies itself since the root cause can often be extremely difficult to troubleshoot due to the inconsistent nature of the issues.

Igor Drobiazko

@elasticio



Igor Drobiazko is a Co-Founder of elastic.io and an experienced open source hacker. He is a PMC member at the Apache Software Foundation and author of two books on web development in the Java programming language. Before co-founding elastic.io, Igor worked at Nokia Siemens Networks, HSBC, and NTT docomo.

“We run Kubernetes on the Google Cloud Platform (GCP), so our experience is based on this particular combination. I wouldn’t say there is one biggest mistake, but certainly a mistake would be...”

Not taking into consideration when outlining the project plan that there most likely will be many workarounds that you’ll have to come up with to get everything up and running the way it should.

Not planning enough time “just in case” may lead to the transition to Kubernetes taking much longer than anticipated, putting the whole project in jeopardy because you cannot deliver on time. For example, during the transition, we had to realize that the default approach with disks and volumes management in GCP and Kubernetes is not fit for a serious enterprise application solution, because it has the so-called “single point of failure” in some cases. So, we had to create a workaround for this and implement our own replication/synchronization schema. Or take such a simple thing as labels: In Kubernetes, labels are limited to 64 characters, which was not the case in Marathon, which we used to have. This, again, made us implement a workaround.

Also, Kubernetes has quite an advanced yet complicated network layer, so it was hard to implement custom network architecture (e.g., certain and unchangeable outgoing IP address) and diagnose network issues (e.g., trace TCP packets that got lost).

‘Five Eyes’ Countries Attribute APT10 At Chinese Intelligence Service

Proactively Addressing New Cyberthreats in Healthcare

TOP SECRET: The New Cyber Threat

Upcoming Webinars »

There are no upcoming webinars at this time.

Download Free eBook



CISO/Security Vendor Relation



provides a number of great features and makes and managing, for example, and now it certainly makes life easier and more convenient.

Eric Bailey

@Mosaic451



Eric Bailey is Chief Information Officer at Mosaic451, a cybersecurity service provider and consultancy with expertise in building, operating, and defending some of the most highly secure networks in North America.

“One of the most overlooked issues while deploying a Kubernetes cluster happens to involve its most important service, the key/value store of the cluster: etcd.”

Ensuring the integrity of etcd on a production cluster could mean the difference between rogue ports opening to critical services exposing sensitive information, or a safe and secure cluster. Role-based access controls are sometimes difficult to put in place, while getting them correct could take time upfront. However, it would save significant time and effort in the long term. A properly set up cluster is a rather large force multiplier for a company looking for scalability and reliability.

Chai Bhat

@VoltDB



Chai Bhat is the Director of Product Marketing at VoltDB.

“Kubernetes has quickly become the most popular container orchestration solution with DevOps and Site Reliability Engineers...”

Kubernetes, however, was designed for stateless web apps that can easily spin up new interchangeable instances in case of node failure or scale-out. On the contrary,



Recent Security Boulevard Cha

Cloud, DevSecOps and Network S
Together?

Security-as-Code with Tim Jeffers
Barracuda Networks

ASRTM with Rohit Sethi, Security

Deception: Art or Science, Ofer Isr
Networks

Tips to Secure IoT and Connected
DigiCert

Industry Spotlight »



5 Ways Hackers K
Sandbox — and V
About It



It's Beginning to L
Like Christmas, w
Cyberattacks Lur
Data Store



The Rise and Fall
Security Technol

Top Stories »



‘Five Eyes’ Count
APT10 Attacks to
Intelligence Servi



Researcher Drop:
Windows Zero-Da

get rid of their monolithic architecture and the database data they store, sacrifice the agility at the cost of reliability that Kubernetes offers. Hence, orchestrating SQL databases in Kubernetes has been a challenging proposition. Operational database systems store state data and can't just be spun up or down on a moment's notice.

Here are a few considerations that database professionals should account for while transitioning to Kubernetes:

1. Whether to utilize Kubernetes clustering or database clustering
2. How to maintain data consistency and reliability

Ben Bromhead

@benbromhead



Ben Bromhead is the CTO at Instaclustr, which provides a managed service platform of open source technologies such as Apache Cassandra, Apache Spark, Elasticsearch, and Apache Kafka. Prior to co-founding Instaclustr, Ben worked as an independent consultant developing NoSQL solutions for enterprises.

“One less understood aspect of transitioning to Kubernetes...”

As relatively straightforward as the container orchestration solution is to get started with once deployed, Kubernetes has limitations as to:

- How persistent data/state is stored
- How it understands different databases

For example, Kubernetes is unable to determine whether you're using a leader/follower database cluster, a single database instance, or a sharded leader-leader infrastructure. These limits can make it challenging to run and manage databases on Kubernetes, and it means that those transitioning to Kubernetes must either build scripts in order to properly operate their databases on Kubernetes, or utilize a Kubernetes operator designed for the task. In some cases, tools are available to mitigate this challenge. For those hoping to use the Apache Cassandra database on Kubernetes, Instaclustr and partner contributors have recently made a free and completely open source Cassandra operator for Kubernetes available on GitHub.

Oleg Atamanenko



Emergency Patch
Vulnerability in Ir
Explorer



Oleg Atamanenko, Lead Platform Developer at Kublr, is a certified Kubernetes administrator and author of cluster autoscaler support for Azure (based on VMSS). Oleg's experience spans more than 13 years, and he lives all things Docker, Kubernetes, AWS, agile methodologies (Scrum, Kanban), and is versed in DevOps languages (Go lang, Java/Scala, bash, JavaScript/TypeScript).

“There is a common misconception that Kubernetes is a fully fledged container orchestration tool, the silver bullet to scaling containers...”

The truth is, Kubernetes is ‘just’ a framework, a great one, but a framework nonetheless. It solves a lot of problems, but there are additional costs that IT needs to consider when introducing Kubernetes in the development and deployment process. While open source is free, making it work isn't. IT must be aware and be prepared to handle the additional complexities that come with a Kubernetes implementation.

Additionally, Kubernetes' quarterly releases may be more frequent than an organization's deployment schedule. Some companies still deploy one or two times a year only, especially in the public sector. That translates into an additional burden for IT operations that need to support a certain Kubernetes version with no added benefit.

Another aspect that is often overlooked is that to truly reap the benefits of Kubernetes, additional business and development process improvements are needed (CI/CD, automated testing, and other best practices). Kubernetes is a great framework, and anyone seeking to scale their container deployments should look into it. However, IT must understand the implications and ensure that they have the resources and skill set in house. If they don't, they should look for a vendor who solves all key operational challenges they aren't equipped to handle.

Recent Articles By Author

- 50 Useful Kubernetes Tutorials for IT Professionals
- 50 Useful Kubernetes Tutorials for IT Professionals
- Introducing Threat Stack's New Podcast: “Your System Called”

ANALYTICSAPPSEC CISO CLOUD DEVOPS GRC IDENTITY INCIDENT RESPONSE IOT / ICS THREATS / BREACHES

*** This is a Security Bloggers Network syndicated blog from Blog – Threat Stack authored by Mike Broberg. Read the original post at:
<https://www.threatstack.com/blog/20-developers-and-kubernetes-experts-reveal-the-biggest-mistakes-people-make-during-the-transition-to-kubernetes>

Containers, Orchestration & Security, Kubernetes Security, Securing Evolving Infrastructure, Security Research & Strategy

← XKCD, History Department

Cylance vs. Sality Malware →

Security Boulevard Comment Policy

Comments are moderated

0 Comments

Security Boulevard

1

Login

Recommend

Tweet

Share

Sort by Best

Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

ALSO ON SECURITY BOULEVARD

126 Arrests: The Emergence of India’s Cyber Crime Detectives Fighting Call

1 comment • 2 days ago

DO IT FOR THE HOPE

Kevin McCoy — "tthat" ???

10 Simple Tips to Protect You from an Email Hack | Avast

1 comment • 12 days ago

Happy Faces —
<https://uploads.disquscdn.c...>

What The Hack 2018?

1 comment • 12 days ago

Roy Betts — The reference to the U.S. Postal Service and Informed Delivery is totally inaccurate and false. The Postal

5 Privacy Tools to Keep Your Data Safe & Secure During the Holidays

2 comments • 11 days ago

Conleth Hammond — Thanks for this list of tools!

[ANALYTICS](#)[APPSEC](#)[CISO](#)[CLOUD](#)[DEVOPS](#)[GRC](#)[IDENTITY](#)[INCIDENT RESPONSE](#)[IOT / ICS](#)[THREATS / BREACHES](#)

SECURITY

Home of the Security Bloggers Network

Join the Community

[Add your blog to Security Bloggers Network](#)

[Write for Security Boulevard](#)

[Bloggers Meetup and Awards](#)

[Ask a Question](#)

Email:
info@securityboulevard.com

Useful Links

[About](#)

[Media Kit](#)

[Sponsors Info](#)

[Copyright](#)

[TOS](#)

[Privacy Policy](#)

Other Mediaops Sites

[Container Journal](#)

[DevOps.com](#)

[DevOps Connect](#)

[DevOps Institute](#)

Copyright © 2018 [MediaOps Inc.](#) All rights reserved.



Our website uses cookies. By continuing to browse the website you are agreeing to our use of cookies. For more information on how we use cookies and how you can disable them, [please read our Privacy Policy](#).

[I Accept.](#)