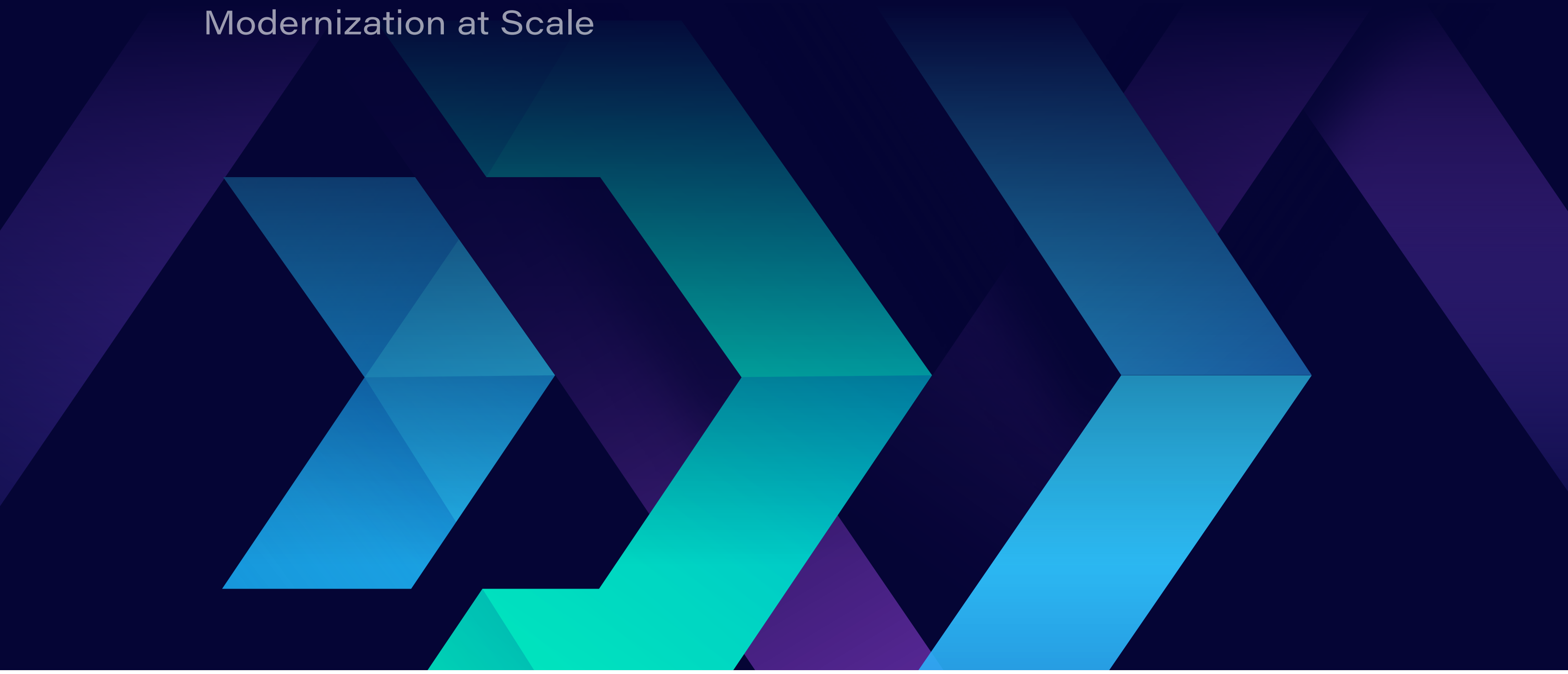
DZone
A DEVADA MEDIA PROPERTY

Kubernetes in the Enterprise Trend Report

Modernization at Scale



BROUGHT TO YOU IN PARTNERSHIP WITH

Couchbase Sysdig influxdata weaveworks

Table of Contents

PRODUCTION

Billy Davis Production Coordinator

Naomi Kromer Sr. Campaign Specialist

MARKETING

Susan Wall CMO

Aaron Tull Dir. of Demand Generation

Waynette Tubbs Dir. of Marketing Comm.

Colin Bish Member Marketing Spec.

Suha Shim Acquisition Marketing Mgr.

Cathy Traugot Content Marketing Mgr.

Melissa Habit Content Marketing Mgr.

Kate Lloyd Customer Marketing Mgr.

Lisa Cronk Graphic Design Specialist

Blake Ethridge Community Manager

EDITORIAL

Mike Gates Senior Editor

Kara Phelps Editorial Project Manager

Tom Smith Research Analyst

Jordan Baker Publications Associate

Sarah Sinning Staff Writer

Andre Lee-Moye Content Coordinator

Lauren Ferrell Content Coordinator

Lindsay Smith Content Coordinator

Charleigh Smith Content Coordinator

Peter Connolly Content Coordinator

BUSINESS & PRODUCT

Terry Waters CEO

Aaron Bailey General Manager, Media

Jesse Davis EVP, Technology

Kellet Atkinson Media Product Manager

SALES

Tevano Green Sr. Account Executive

Alex Crafts Key Account Manager

To sponsor a Trend Report:
Call: **(919) 678-0300**
Email: **sales@devada.com**

ORIGINAL RESEARCH

Highlights & Introduction

By Kara Phelps, Editorial Project Manager at DZone

This year marks the fifth birthday for Kubernetes, today's de facto open source container orchestration system for much of the tech industry. It's now the largest proof of concept for open source development. In just five years, believe it or not, Kubernetes has become virtually indispensable to many aspects of modern software. A thriving ecosystem of related tools has grown around it. Some of the largest modern tech companies now rely on Kubernetes (also known as K8s) to scale quickly and to provide stateless, flexible infrastructure.

Still, enterprise and legacy organizations frequently struggle with K8s adoption and long-term maintenance. Strong enough security protocols are an ongoing concern, as well. Teams often abandon Kubernetes due to its steep learning curve and the complexities of installation and daily operation. The tech world needs more developers to be highly skilled in K8s.

For all these reasons and more, the DZone team decided to publish our first-ever Kubernetes Trend Report with a specific focus on applications in the enterprise environment. We're exploring the challenges of container orchestration at the enterprise level, diving into a few ways to automate security in the Kubernetes pipeline, and examining enterprise Kubernetes deployment from the ground floor up.

Kosmas Pouianou opens up the report with his article that covers the history of K8s and containers as well as musings on their future, including examples of use cases from the enterprise world.

In "Automating Open Source Security in Kubernetes Throughout the DevOps Pipeline," Shiri Ivtsan addresses one of the most overlooked challenges encountered when integrating K8s into the software development lifecycle: open source security.

TREND PREDICTIONS

- Kubernetes has seen staggering growth in the five years since its inception, and there are no signs of its popularity waning anytime soon.
- Developers with advanced skills in Kubernetes will be in increasingly high demand.
- As the enterprise world begins to modernize legacy systems with the help of Kubernetes, security will become an even more important part of the K8s development and deployment pipeline.

Adi Polak and Idan Levin tackle the nitty-gritty details of an organization's first K8s deployment in their article, "Deploying Kubernetes in an Enterprise Environment." They review requirements for Kubernetes product solutions at the enterprise level; the necessary composition of the K8s development team; and current best practices for securing virtual machines, managing access, monitoring, and more.

DZone Research

We also have the results from a recent reader survey on Kubernetes and containers. In the Key Research Findings later in this report, we're sharing what we've learned about how and why software organizations now use K8s. Usage rates have exploded from 2018 to 2019, and we look at the usage rate through the lens of organization size. We also delve further into the data to explore how many containers are typically being run in production, as well as the most common reasons for using container orchestration tools.

We delve into our reader survey data to explore how many containers organizations typically run in production.

Finally, DZone's research analyst Tom Smith shares his Executive Insights on the State of Kubernetes in the Enterprise. Tom has gathered these Executive Insights from interviews with IT executives and C-levels at 22 tech companies. You can turn to the end of this report to read their observations on the current landscape and their predictions for the future.

Kubernetes has seen staggering growth in five short years, and its story is far from over. There are no signs of its popularity waning anytime soon. Thank you for your interest in bringing K8s to the enterprise as part of the next phase of its evolution — and thank you for downloading this report to explore further. Drop us a line and let us know what you think of it; we're always glad to hear your feedback.

OPINION

Kubernetes in the Enterprise

By Kosmas Pouianou, Data Engineer at SAP

Since 2014, the aptly-named Kubernetes (the name means "governor" in Greek) has become the default in container orchestration systems, backed by some of the largest companies in the industry, with a vibrant ecosystem of open source and commercial systems built around and on top of it. What about the industry landscape, however? What impact will it have and will K8s revolutionize the traditional enterprise data center altogether?

To tackle these questions, we first need a basic understanding of both the underlying technology and enterprise-specific challenges to tackle.

TREND PREDICTIONS

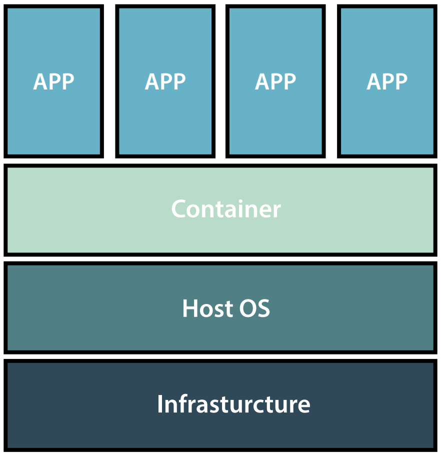
- As the Kubernetes ecosystem evolves, a positive feedback loop emerges, resulting in increasingly sophisticated technology.
- Kubernetes is increasingly making headway into the enterprise world, and it will be interesting to see how this growth continues and how the ecosystem will adapt and evolve in turn.

What Are Containers and Why Do They Matter?

To put it simply, a container is a piece of software that packages code and its dependencies (system tools, runtime, libraries, binaries, etc.) and runs it on a host machine's OS kernel in an isolated environment.

This offers several benefits, for example:

- Portability: Containerized software can run on any infrastructure consistently. Tired of hearing the old "but it works on my laptop!" excuse for production bugs? Portability goes a long way to solving this problem and is one of the pillars of the modern cloud.
- Resource Efficiency and Speed: One of the key features of containers is that, unlike virtual machines (VMs), they don't virtualize the hardware; rather, they just virtualize the OS, allowing multiple containers to share OS resources. Essentially, this means many more containers can run simultaneously on the same machine, considerably lowering costs. At the same time, containers are very fast to start up. If you've ever heard of the "serverless" buzzword, this is what makes it at all possible.



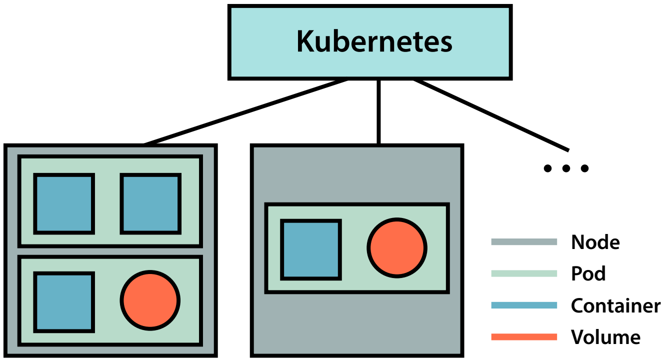
While containers on their own bring a lot to the table, the industry-changing benefits only become apparent when one takes the next logical step: container orchestration. This is exactly where K8s comes in.

What Is Container Orchestration and Why Is It Important?

Modern applications, at least significantly large ones, are often no longer monoliths; rather, they consist of several loosely coupled components that need to communicate and work in tandem. For example, an app might use a service for authentication, another for ingesting data from social media streams, and yet another for serving an analytics dashboard. Such services can be run in separate containers, allowing developers to release, deploy, and scale such services independently. This offers a nice separation of concerns, enabling faster release cycles for key components, as well as efficient resource allocation.

Container orchestration solves several challenges that arise from such an architecture. For example:

- Automated deployment and replication of containers
- Load balancing
- Rolling updates (updating containerized apps with no downtime)
- High availability: When a container fails, its replicas continue to provide service
- Self-healing: Allows devs to restart failing containers, kill containers that fail to respond, or replace containers when a node dies
- Secure communication between containers



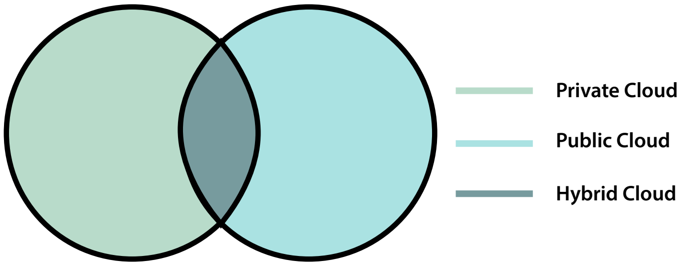
It’s obvious that these features are among the pillars of the modern cloud, and partly explain why K8s has become ubiquitous. It is equally easy to see why this approach is a perfect fit for stateless apps in particular. But what about the needs of big enterprise systems?

Tackling Challenges in the Enterprise

Managing State: Let’s address the elephant in the room straight away. While not an enterprise-specific challenge, it is an important one. Stateful applications, like databases, caches, and message queues, face challenges in terms of portability, since state needs to be maintained whenever a container starts, stops, or is replicated. This is particularly challenging in a distributed or even multi-cloud environment.

K8s tries to address this mainly through Volumes, Persistent Volumes, and StatefulSets. In practice, all these options are great to have and cover many scenarios; but, for the time being, there are still many that they do not, and the sheer complexity of containerizing stateful apps, in general, often outweighs the benefits in production scenarios. The question of managing storage and containerizing stateful apps is a very hot topic, and there's a lot of effort being put in this direction in the industry (e.g., Ceph, Rook, KubeDirector, KubeDB, Red Hat's Operator Framework).

Security: This is a big deal in the enterprise world. Despite their many advantages, containers do not offer the same level of isolation as VMs. Multi-tenancy, in particular, can be a challenge. Again, there is a lot of effort being put into making containers more secure; a great example being Google open-sourcing gVisor in a bid to bring better isolation to containers — and it integrates nicely with K8s.



High Performance Computing (HPC): Enterprise data centers typically run a variety of workloads on different types of servers, for example, GPU machines meant to run intensive compute operations like ML/AI pipelines. To address this, K8s uses taints and tolerations to ensure that pods are scheduled into appropriate nodes (the physical or virtual machines where containers run).

This approach essentially allows workloads to run on the appropriate infrastructure and can be of use in other cases, for example, running workloads on machines within a DMZ.

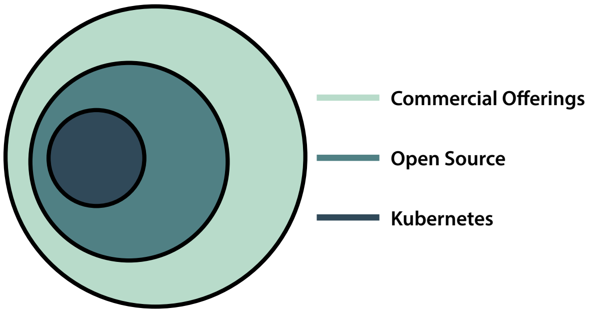
Multi-cloud: Enabling hybrid/cloud deployments and avoiding vendor lock-in are key requirements for the modern enterprise. This poses significant technical challenges which cannot be addressed by a simple tool, but typically require a combination of technologies and architectural approaches. This is one of the reasons why we've seen a growth in enterprise K8s offerings, such as OpenShift, Docker Enterprise, and Google's Anthos.

An Open Source Success Story

K8s is, today, one of the top open source projects, operating under the Cloud Native Computing Foundation (CNCF), itself part of the Linux Foundation. The CNCF acts as an umbrella organization to K8s and is backed by some of the largest companies in the industry, like Apple, Microsoft, Google, Amazon, SAP, Oracle, and many others. As a result, a vast ecosystem of open source technologies has evolved around K8s. This includes a wide variety of technologies from monitoring solutions, like Prometheus and container runtimes like Containerd, to package managers, like Helm. Many of the biggest players in the industry are thus incentivized to take part in shaping the future of cloud computing by contributing to the CNCF and, in turn, leveraging the ecosystem for their commercial offerings.

Examples in the Enterprise

SAP is a company that faces many of these daily enterprise challenges, such as the need for large-scale, heterogenous multi-cloud and hybrid solutions. This is where Gardener comes in: an open source project offering operation of K8s clusters as a service, on various cloud providers, at scale. Internally, SAP's business platform, SAP Cloud Platform, leverages Gardener to provision K8s clusters to its customers.



In a similar vein, SAP developed the open source Kyma project on top of K8s to extend and customize cloud and on-premise enterprise applications. The vision for Kyma is to act as the glue between Mode 1 and Mode 2 environments, essentially allowing users to extend their Mode 1 environment with Mode 2 capabilities, without disrupting the existing Mode 1 systems.

Finally, in terms of commercial offerings, SAP leverages K8s for its Data Intelligence cloud service, as well as its on-premise enterprise data orchestration solution, DataHub.

Moving Forward

We already discussed challenges specific to the enterprise and what options K8s and its ecosystem employ to address them. As a takeaway, there are a few points worth reiterating:

- Containers are not the answer to everything, but K8s is probably the default way to manage containerized systems at present.
- Container tech is constantly evolving and a lot of effort is being put into overcoming current challenges (e.g., containerizing databases).
- As the K8s ecosystem evolves, a positive feedback loop emerges, resulting in increasingly sophisticated technology.
- Most big tech players are directly involved in enriching the ecosystem and building commercial offerings on top of it.

Considering these points, it is clear we are currently at an exciting point in the evolution of cloud native technologies. As for Kubernetes, it is increasingly making headway into the enterprise world. It will be interesting to see how this growth continues and how the ecosystem will adapt and evolve in turn.

SPONSOR OPINION

Autonomous Couchbase on Kubernetes With Cloud 2.0

By Anil Kumar, Director of Product Management at Couchbase

Two years ago there were many competing standards for container orchestration. Today Kubernetes has emerged as the clear frontrunner and has become the de facto standard to power the next phase of cloud technology.

Kubernetes Will Power Cloud 2.0

Looking forward a year, the ecosystem around containers will continue to grow, and soon Kubernetes will be more than just the orchestration layer — it will become the operating system for an intelligent, business-driven Cloud 2.0 that incorporates true multicloud strategies.

TREND PREDICTIONS

- Kubernetes will become the operating system for Cloud 2.0.
- Cloud 2.0 will incorporate true multicloud strategies.
- Containerization of stateful applications will go mainstream.

Cloud, especially Kubernetes, is fantastically successful for deploying and testing stateless applications. But underpinning the system, you need a database to drive the applications and provide operational and analytical insight. Stateful applications have the most resistance to change and are always the last to migrate to new technologies.

Couchbase prepares you for Cloud 2.0 with the Couchbase Autonomous Operator for Kubernetes. It runs as an internal database as a service and reduces operational complexity by up to 95% by implementing best practices and running Couchbase as an autonomous, fully managed stateful database application next to other microservices applications, all on the same Kubernetes platform.

Leverage Cloud Portability Across Platforms and Providers

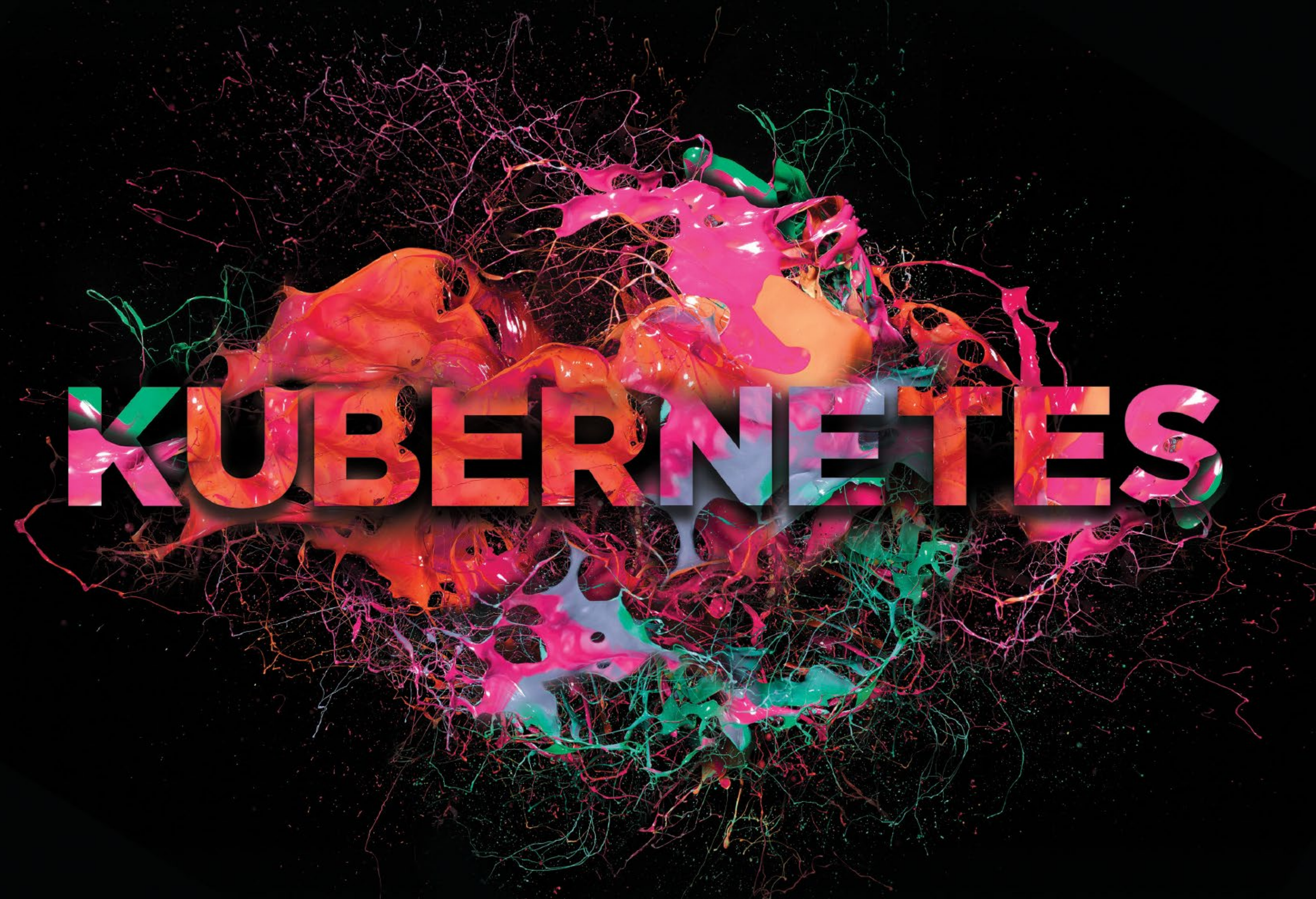
Today's enterprises depend on a mix of private and public clouds, and need frictionless data portability in various cloud platforms. Operator for Kubernetes provides easy portability across private, public, and multiclouds.

Key Features of Couchbase

1. Full-featured: Native integration with Kubernetes provides a comprehensive NoSQL database that supports critical applications with unparalleled performance.

2. Deploy at will: Couchbase doesn't force you to choose between on-premises, private cloud, or a specific public cloud deployment. You can easily deploy Couchbase within a managed private or public cloud to maximize flexibility, customizability, and performance.

3. Use what you know: Couchbase has developed strategic partnerships with the most popular enterprise providers, including Red Hat OpenShift Container Platform and cloud-managed Kubernetes services on AWS (Amazon EKS), Azure (Azure AKS), and Google (Google GKE). As cloud vendors build more ways to integrate with their container platforms, Couchbase makes it easier to take advantage of their latest advancements.


KUBERNETES

OPINION

Automating Open Source Security in Kubernetes Throughout the DevOps Pipeline

By Shiri Ivtsan, Senior Product Manager at WhiteSource

Kubernetes adoption is at an all-time high and doesn't seem to be slowing down. As container usage continues to soar, this popular orchestration framework helps teams to deploy and scale containerized applications at the speed of DevOps.

However, as is often the case when attempting to speed up development and delivery, security is slow to join the party. Enthusiastic users tend to forget that integrating Kubernetes into the software development lifecycle also introduces a new set of security concerns, and one of the top issues that is often overlooked is open source security.

TREND PREDICTIONS

- Technologists will begin using a next generation toolbox of automated solutions to secure their software projects from the earliest stages of development.
- Kubernetes will continue to play a prominent role in development because of its ability to easily integrate with automated security tools.
- Organizations are going to begin bolstering their DevSecOps efforts with automation to gain an edge.

Kubernetes and the Challenge of Open Source Security

Nearly 97% of developers rely heavily on open source components, which compromise 60-80% of today's software applications. Containers are no exception to this statistic, and Kubernetes, home to one of the most active open source communities around, is purely open source, including its underlying infrastructure. In practice, this means that open source security is the basis for Kubernetes security.

Considering the fact that Kubernetes and the containerized environments that rely on it are very much an open source ecosystem, it's important to understand the challenges that open source components with known vulnerabilities pose to organizations using Kubernetes.

As open source usage has gone mainstream across industries of all sizes and verticals over the past few years, the number of known open source security vulnerabilities has risen exponentially from year to year. This requires software development organizations to start paying attention to open source security and to do their best to address known open source security vulnerabilities.

Open source vulnerabilities management requires a different set of processes and tools than securing proprietary or commercial code. When open source security vulnerabilities are discovered, they are quickly published for the whole world to see — users and malicious actors alike. It's up to the users to stay on top of which open source components and versions they are using and make sure that the components are up-to-date and secure before a hacker finds a flaw.

Tracking which open source components you are using is no easy feat considering the volume of open source code in today's software, and, to make matters even more complicated, the decentralized nature of the open source community means that known open source vulnerabilities are published across a number of community issue trackers and advisories, rather than in one centralized location. That makes keeping track of known open source vulnerabilities and remediating the vulnerable components in your products an impossible feat to carry out manually, especially at scale.

If that's not daunting enough, there is also the issue of the tangled web of dependencies in open source libraries. Most open source components are dependent on other open-source components. Tracking those dependencies is critical to keeping your entire codebase secure since a vulnerability in an underlying component can impact all other software that is built on top of it.

Covering All of the Layers: Integrating Security Into the Kubernetes Pipeline

Ensuring open source security in your Kubernetes usage requires integrating a DevSecOps approach throughout the software development lifecycle — from the early stages of research and development, as well as understanding which architecture and components to use, through the build stage, and all the way up to deployment. I've mapped out the main stages when known open source vulnerabilities should be addressed to ensure a secure deployment.

Research and Coding: Shifting (Farther) Left

This is when the magic starts to happen. Developers get a task and start searching for available open source components in one of the popular (and free) community repositories. This early step of selection requires developers to be aware of which open source components have known vulnerabilities, so as to avoid using them in the first place.

As they continue to code away, developers will probably continue adding open source dependencies. Here, too, it's important that they know exactly what they are using as building blocks in their project, and make sure components — dependencies included — are vulnerability free.

> Ensuring open source security in your Kubernetes usage requires integrating a DevSecOps approach throughout the software development lifecycle.

An ounce of prevention is worth a pound of cure, as they say.

This is a great example of the shift left approach, when you address security vulnerabilities as early as possible — in this case, even before they are added to the image registry. Keeping your Kubernetes processes secure can, and should, start before code even gets to the containerized environment. This saves a lot of time and money that would otherwise be spent fixing issues later on, closer to delivery dates, when remediation is a bigger and far more expensive task.

The Build: Security Gates

Failing the build based on automated code functionality tests is considered best practice today, and rightly so. In keeping with the shift left approach of doing everything you can to ensure code is secure before it even arrives at your containerized environment, we recommend failing a build when a security vulnerability is hiding in your code.

Integrating vulnerability scanning into your CI processes ensures that open source security vulnerabilities are blocked from entering your project, and provides developers, DevOps, and security teams with an easy, seamless process of weeding out security issues before they become costly at a later point in the SDLC.

Storing Kubernetes Images: Keeping the Image Clean

A container image registry is a service that stores container images. It's hosted either by a third party or as a public/private registry — Docker Hub and Quay are two popular registries. When the image is stored in the image registry, new vulnerabilities are often discovered in open source components that were previously found to be safe. Another scenario that unfortunately occurs often is that after the build you get additional third-party images that were not scanned for vulnerabilities.

That's why it's important to continuously scan your code throughout the DevSecOps process while it's stored, even though the code is not actively being "developed." This also provides one more security gate before the image is deployed to your test or production environment.

> The development lifecycle proceeds at an increasingly fast pace.

Deploying to Kubernetes: Security That Doesn't Stop

The development lifecycle proceeds at an increasingly fast pace. While continuously tracking your software projects for vulnerabilities is essential, you still need to scan for vulnerabilities in the deployment stage to ensure that you're not deploying images with known vulnerabilities.

Deployment is your last security gate before your image or container starts getting production traffic from real users. This is your final chance to stop projects with known vulnerabilities from being deployed.

At this stage, enforcing policies that block any vulnerable open source components in your code is of the utmost importance, and automated policy enforcement will make the task that much easier. The Kubernetes admission controller, for example, is a good tool that allows you to enforce such rules.

The Future Will Be Automated: Baking Security Into the Entire Kubernetes Lifecycle

Currently, most of the security measures that we described to address known open source vulnerabilities when working with Kubernetes orchestration are implemented manually, if at all.

Considering the speed of software development and scale of open source code being used in modern applications, manually tracking and remediating vulnerable open source components in a project is extremely time consuming and downright impractical at scale. Not only that, the margin for human error is far too wide and its implications can range from costly to disastrous.

> The margin for human error is far too wide and its implications can range from costly to disastrous.

The good news is that all of these processes can be easily automated, and, looking to the future, organizations are going to begin bolstering their DevSecOps game with automation to gain an edge.

The scale of open source components' usage combined with the adoption of shift left processes are putting more weight on the shoulders of developers when it comes to security responsibilities. Keeping up with these new responsibilities while organizations continue to speed up development processes is impossible without automating security.

Going forward, we will see developers, DevOps, and security teams using a next generation toolbox of automated solutions to secure their software projects from the earliest stages of development, all the way to ensuring that vulnerabilities are automatically blocked at security gates.

Kubernetes is a prime example of an environment where automated security tools can be seamlessly integrated by all of the relevant teams to ensure that development proceeds at the breakneck speed of DevOps, all without compromising on security.

SPONSOR OPINION

Securing and Monitoring Kubernetes Requires a Data-Driven Approach

By Pawan Shankar, Product Marketing Manager, Sysdig

Kubernetes has taken the container ecosystem by storm. Although many enterprises are scaling Kubernetes in production, they are facing increased complexity for managing and scaling operations. This is highlighted by the increase in security attacks, vulnerabilities, and exposures over the last few months. There are four factors that consistently contribute to operational challenges for Kubernetes in the enterprise:

- **Distributed microservices:** Applications are now being designed for microservices. An order of magnitude more components are communicating and distributed in containers or serverless functions across multiple cloud providers or on-premises.
- **Complex attack vectors:** Attackers may access compromised API access keys, leverage a vulnerability in base images (used for containers, outsourced libraries, or in serverless function code), or take advantage of vulnerabilities inside the orchestrator settings to reach services that contain sensitive information.
- **Ephemeral services:** The ephemeral nature of containers (95% are said to live less than a week) combined with their opaque characteristics and the massive volume of data, makes wading through a plethora of security or performance issues like finding a needle in a haystack.
- **Security seen as the bottleneck:** By not including security in the approach early and collaborating often, there is greater risk for exposure into the app dev infrastructure.

TREND PREDICTIONS

- ► Greater Kubernetes adoption
- ► Complexity migrates to container production phase
- ► Increase in security attacks and vulnerabilities
- ► Need for simplicity to manage and scale

Containers demand a new approach to security, which is rarely a niche, single function tool. Data is key to understanding what is happening in the ephemeral container realm, and insight is key to securing these dynamic environments. Ultimately, visibility into both security and performance data is critical to operate reliable containers at scale.

At Sysdig, we use data to solve Kubernetes visibility and security as a converged problem. We offer the only unified approach to security, monitoring, and forensics in Kubernetes environments. Sysdig delivers:

- A unified overview of health, risk and performance of any application
- Image scanning and vulnerability prevention
- Compliance and audit
- Full stack visibility, capacity planning and performance management
- Runtime security, troubleshooting and forensics

Learn more or schedule a demo at www.sysdig.com.

OPINION

Deploying Kubernetes in an Enterprise Environment

By Adi Polak, Senior Software Engineer and Developer Advocate at Microsoft
By Idan Levin, Chief Architect for MDATP at Microsoft

So, picture this: you can't stop talking about Kubernetes. After some initial discussions, your company's stakeholders have finally given you the green light to adopt it. You are driving a technological change in an enterprise environment, which is very different than running it in a startup or a small company. You are accountable for a huge product's success. How do you proceed? What do you do? We've got you covered. Just continue reading.

- If you are totally new to Kubernetes, we recommend going over the basic core concepts before reading the article.

TREND PREDICTIONS

- ► Kubernetes changes every day. Hence, you will need a team in place. You can hire or create your own team of Kubernetes experts, but it takes time to train or hire the right people with the right skillsets and passion.
- ► Various managed Kubernetes solutions will continue to target solving security challenges, operability challenges, and much more. Take all solutions into consideration, as they might save you time and effort on this long journey of deploying Kubernetes in the enterprise.

Introduction

Kubernetes is a great container orchestrator, with out-of-the-box luminary capabilities like resource management, deployment management, automatic updates, self-healing, and many more. Although Kubernetes itself is a great step toward a full-blown, enterprise-grade production environment, it is not enough.

In this article, we will cover:

- What enterprise grade Kubernetes product solution requirements are and how to target them.
- What the team should look like, e.g., should you share cluster resources with other teams?
- What to do to secure virtual machines, access management, monitoring, and more.

Team of Experts

First things first, build your team. Kubernetes is far from a simple platform; with great power comes great complexity. To run it properly, you'll need a team (or a v-team) of experts, responsible for the "infrastructure." Whether you hire experienced Kubernetes engineers externally, or train existing employees, you should invest a significant amount of resources in building the team's skillset. Team members should read Kubernetes books, participate in conferences (like KubeCon), and keep up-to-date with new versions and new functionality — even deprecated ones! Team members' main goal should be to abstract away complexities for the rest of the group. They should be responsible for provisioning the clusters, hardening them, monitoring them, responding to the infrastructure related incidents, and more. Most organizations call them DevOps or Site Reliability Engineers (SREs).

Cluster Resources: To Share or Not to Share?

Whether you run Kubernetes on-premises or in the cloud, one of the main questions you should ask is: should we share Kubernetes clusters between multiple teams or separate them?

Well, Kubernetes was built in a way that allows it to be shared by multiple teams. It has built-in Role-Based Access Control (RBAC) that allows you to give only the required permissions to each entity. It also has the Namespaces ability, which creates virtual clusters that are backed by the same physical one. Sharing a cluster enables better auto-scaling for saving COGs (cost of goods), it has a smaller operational cost, better support network policies, and, if you're not doing in-cluster TLS, it actually results in better performance.

So, in answer to the question above: share!

Securing Virtual Machines

Conditional to how you provision Kubernetes, you either have access to both the master and worker nodes (e.g., running your own self-managed Kubernetes) or just the workers (e.g., using a cloud-managed Kubernetes engine). Either way, you must secure them.

- Always use up-to-date operating systems and patch when needed.
- Make sure the security updates are automatically installed.
- Deploy antivirus and EDR (Endpoint Detection and Response) solutions.
- Deploy a vulnerability scanning agent.
- Reboot daily, or at least weekly.
- Export security and audit logs to a centralized server (outside of the cluster) for detection and response.

Kubernetes doesn't (yet) provide an easy way to manage the underlying VMs. However, there are two approaches you can take to implement the above:

1. Use a configuration management tool like Chef, Puppet, Ansible, or Terraform.
2. Deploy a privileged Kubernetes DaemonSet that uses hostPath volume to run commands on the host (e.g., using chroot.)

One thing to consider when choosing between these alternatives is cluster auto-scaling. With Option 2, you don't have to do anything; auto-scaling will just work. Option 1, however, requires special handling.

Access Management

No one should ever have standing access (always on) to production environments; you should implement just-in-time (JIT) access and make sure you closely monitor those connections. Avoid using the certificate you received when creating the cluster (the Kubeconfig file), use an identity-based authentication solution like Azure Active Directory or any solution that implements OpenID Connect; a hands-on tutorial and more information are available here.

Monitoring

Anything that runs on Kubernetes, including the underlying virtual machines, needs to be monitored, where basic monitoring includes logs and metrics.

In Figure 1, all the containers write logs to STDOUT, then the container runtime persists these logs to files on the host. Fluentd is used here as the logging layer (DaemonSet) that tails, aggregates, compresses, and sends the log files to a centralized database.

Kubernetes, and almost any other open source product that runs on top of it, is instrumented for metrics exporting with Prometheus. You should therefore consider installing Prometheus in the cluster to scrape all the containers for metrics, aggregate them, compress them, and then export them to a centralized database as well.

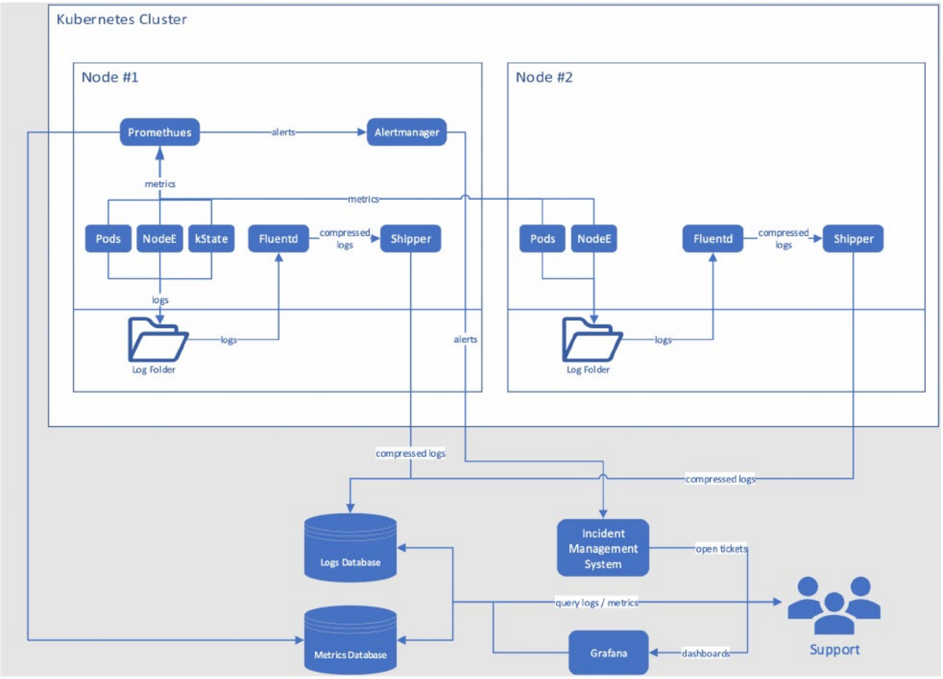


Figure 1

To monitor the underlying virtual machines (VMs), you can use the Prometheus Node Exporter DaemonSet (NodeE in the diagram). For the Kubernetes metrics layer, use the kube-state-metrics exporter. As an example, you can deploy Prometheus' AlertManager, which takes care of things like alerts silencing and grouping. Using it allows you to plug in your favorable incident management tools as well. As a last piece for your monitoring system, you will need a dashboard. As a dashboard for the whole alert and monitoring systems, you can leverage Grafana. There is also Grafana-Kubernetes-app, which is currently free, but keep an eye on the license as it might change.

High Availability and Disaster Recovery

You want to make sure your product will always be able to serve your customers with minimum down time. This is especially important when developing for the enterprise, where there is zero tolerance for underperforming products. For that, you'll need your Kubernetes clusters to be Highly Available (HA) and have a Disaster Recovery (DR) plan intact. It is not an easy task and should be taken seriously.

Here are best practices regarding how to achieve HA and DR on Kubernetes.

Start with making sure that individual clusters are set up correctly, meaning that there are at least three master nodes to survive failures (we need a quorum). And, if possible, use multiple Availability Zones. The fact that the masters are healthy doesn't mean your application is. Therefore:

1. Avoid having only one replica of a pod.
2. Define Liveness and Readiness Probes in every pod.
3. Enforce Resource Requests and Limits in every pod.
4. Use Pod Priority to prevent critical pods from being evicted from nodes.

A good example of a service that should implement the above best practices is the Ingress Controller (a.k.a. the cluster gateway). Since this is the main entry point to your application services, it has to be up and running at all costs.

Even after strictly following all the recommendations above, there are many reasons an entire Kubernetes cluster can suddenly break — anything from a bad cluster upgrade to an entire datacenter catching fire after a lightning strike. That's why you should have at least two Kubernetes clusters in different geographical locations, as this will help you to survive failures stemming from outages and/or natural disasters.

In Figure 2, we see an example of how running two Kubernetes clusters can look on a public or private cloud. Two Kubernetes clusters are deployed in different regions and a DNS Server (in the Traffic Manager diagram) is routing the traffic between the clusters' gateways. We recommend this architecture since HTTP traffic is relativity easy to fail over. Just change the IP for the DNS and you're done. However, there might be more complicated scenarios — for example, when we have CronJobs that should run only on one of the clusters, or Queue Workers reading from a specific Service Bus. Having a proper DR solution for those depends heavily on service architecture. Be aware of this potential bottleneck and work with the group to define the right software solution.

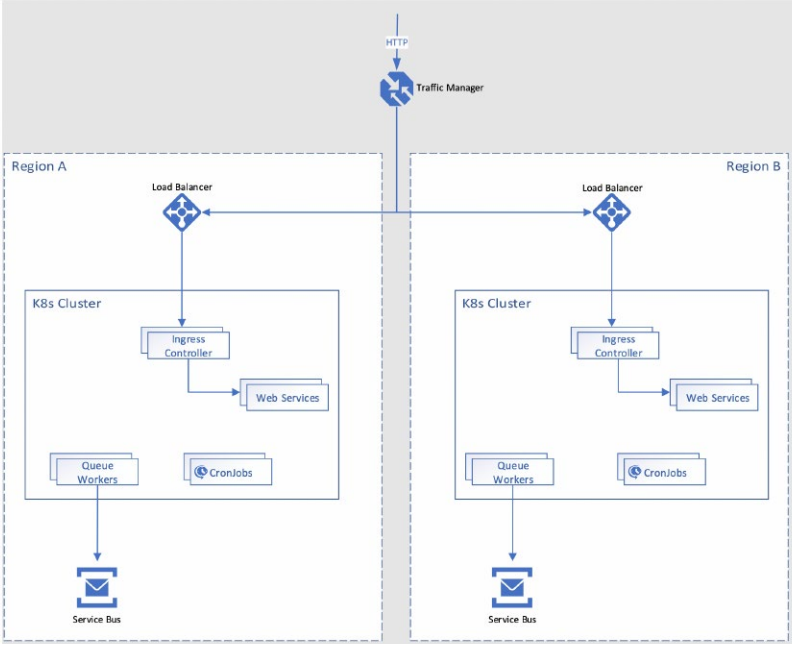


Figure 2

Deployment

Now that the clusters are secured, monitored, and highly available, it's time to deploy your services. Avoid running manual commands on the cluster. Instead, adopt a model where a Git commit to the master branch automatically kick-starts the production deployment. It is best to configure your cluster state in Git, just like your code.

Services running on multiple clusters in different regions might require different configurations to operate. One way to avoid many YAML duplications is by templating the environment variables; this can be done using Helm.

Tip: You can create one shared Helm chart, managed by your Kubernetes experts, for the entire group. This will make your developers' lives easier.

Most applications require Secrets, but, as a general rule, avoid using native Kubernetes Secrets — they are insecure. Instead, use fully managed identity solutions (like AAD POS identity) wherever possible; if you still need to use secrets, keep them safe in a Key Vault (see Azure Key Vault). Lastly, to retrieve them in runtime, we recommend using Kubernetes InitContainer. Don't forget to rotate your secrets every couple of months (as defined in your security policy).

There are many topics that need to be covered when discussing Kubernetes in the enterprise. In this article, we addressed the basics of getting started and formulizing your draft solution. For a deeper, hands-on look at K8s and security-focused topics, follow us on our blogs and social media. We are here for you, so let's discuss your questions, insights, ideas, and concerns. Write us!

SPONSOR OPINION

Building a Development Ready Kubernetes Platform

By Anita Buehrle, Senior Content Lead at Weaveworks

Congratulations on starting your cloud-native journey. Your team has chosen the leading development and deployment framework that provides application portability, agility, and scalability. You began your journey with containers and now you're ready to deploy your container-based application at scale with Kubernetes. But at this point you're faced with a bewildering array of software vendors, cloud providers, and open source projects that all promise painless, successful Kubernetes deployments.

How do you decide where to go from here?

The key to success is a flexible and reproducible cloud native platform that allows you to quickly adopt these new technologies in your infrastructure and to run workloads anywhere: on premise, in public clouds, or even in a hybrid-cloud environment.

"Cloud-native applications increase business agility and speed. But this requires a new runtime platform and environment for operating cloud-native applications reliably, securely, and at scale." Steve George, COO Weaveworks

Weaveworks Enterprise Kubernetes Platform reduces this complexity through automated configuration management and operations tooling. With GitOps configuration management, teams can define a standard installation of Kubernetes and automate the deployment of new nodes following standard templates. Preconfigured cluster templates let developers and operators define apps and update clusters add-ons with security patches, minimizing the YAML mess.

GitOps Configuration Management Automation

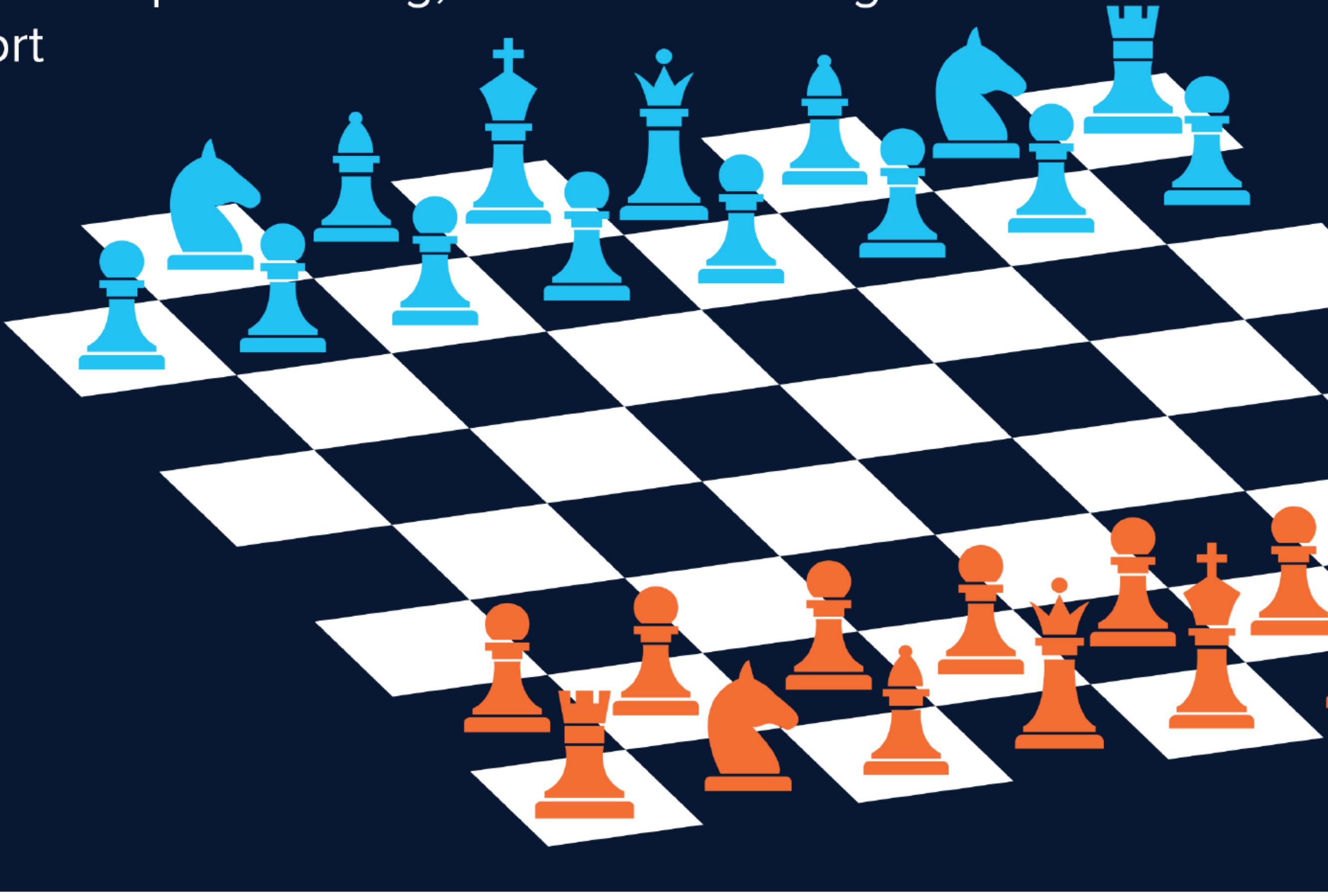
With GitOps at the center of your operational model, application developers and cluster operators can spin up and manage production ready Kubernetes across environments with ease. GitOps can initiate a cluster patch or a minor version upgrade or add and remove cluster nodes all without having to rebuild your entire cluster from the ground up.

When your entire cluster configuration is stored in Git and managed with GitOps, you can reproduce the cluster in a repeatable and predictable way. This brings advantages when you are building test environments and pipelines, and producing clusters for different teams with the same base configuration, or improving your disaster recovery capability.

Weaveworks Enterprise Kubernetes Platform

Increase application delivery at enterprise scale. Reduce the time, effort, and errors to create, update, and manage production ready clusters. Preconfigured dashboards allow you to understand clusters, verify and correct updates, and alert on incorrect states.

Find out more about the Enterprise Kubernetes Platform.



ORIGINAL RESEARCH

Key Research Findings

By Jordan Baker, Publications Associate at DZone

Demographics

For this article, we've drawn upon data from a survey conducted among the DZone member community. In the survey, we asked how respondents' organizations use containers and cloud technologies, and how Kubernetes fits into the picture.

Before we dive into where and how Kubernetes is used, let's go over our respondents' basic demographic information.

- Respondents live in three main geographical areas:
  - 35% live in Europe.
  - 26% reside in the United States.
  - 14% live in South Central Asia.
- Most respondents work for enterprise-level organizations:
  - 23% work for organizations sized 100-999.
  - 22% work for organizations sized 1,000-9,999.
  - 22% work for organizations sized 10,000+.
  - 17% work for organizations sized 1-19.
  - 15% work for organizations sized 20-99.
- Respondents tend to work on two main types of applications:
  - 84% develop web applications.
  - 52% develop enterprise business applications.
- Respondents typically fill one of three main roles for their organization:
  - 29% work as developers.
  - 25% are architects.
  - 20% work as developer team leads.

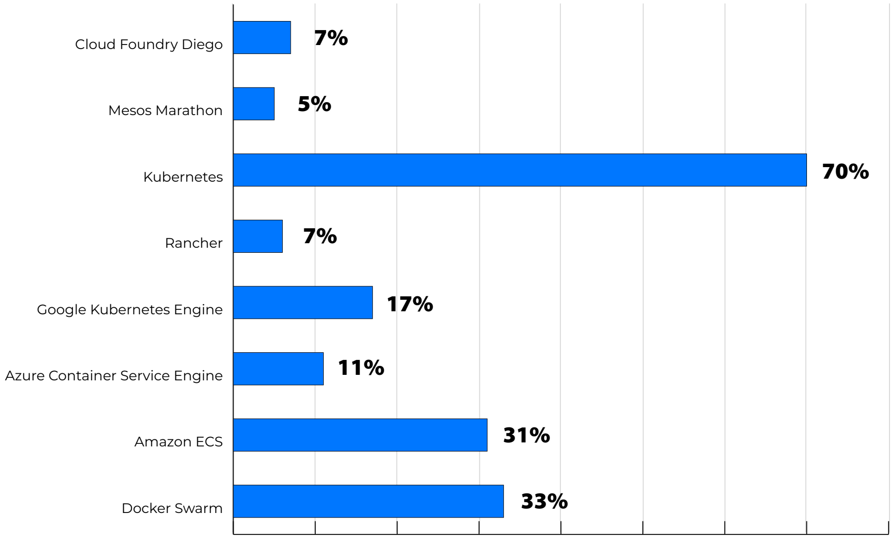
TREND PREDICTIONS

- Due to its open source nature, Kubernetes usage rates among organizations will continue to grow.
- More and more industry leaders will use Kubernetes for container orchestration.
- As developers become more familiar with Kubernetes, enthusiasm for the Kubernetes ecosystem will also continue to increase in the developer community.

- Respondents' organizations tend to use four main programming language ecosystems:
  - 84% use Java.
  - 67% use client-side JavaScript.
  - 48% use Node.js/server-side JavaScript.
  - 41% use Python.

Where Kubernetes Is Used

Figure 1: Which container orchestration/management technologies does your organization use?



Kubernetes (K8s) is a technology on the rise and its use when developing containerized software is almost a foredrawn conclusion. In 2018, 53% of respondents told us that they used Kubernetes for container orchestration; in 2019, this rose to 70%. That's quite the year-over-year increase. As is to be expected with such a large jump in usage, Kubernetes proved rather ubiquitous in certain areas of software development. For one, K8s usage rates among the four main departments in a software development organization (development, operations, DevOps, and QA/Testing) were all within three percentage points of each other. According to respondents, 75% of their organizations use K8s in DevOps, 73% use it in QA/Testing and Operations, and 72% use K8s in development. Additionally, there does not seem to be a preferred cloud hosting strategy among Kubernetes users. 78% of respondents whose organizations use private cloud also use K8s, 76% of respondents whose organization use hosted clouds also use K8s for container orchestration, and 73% of respondents whose organization use local hardware use K8s.

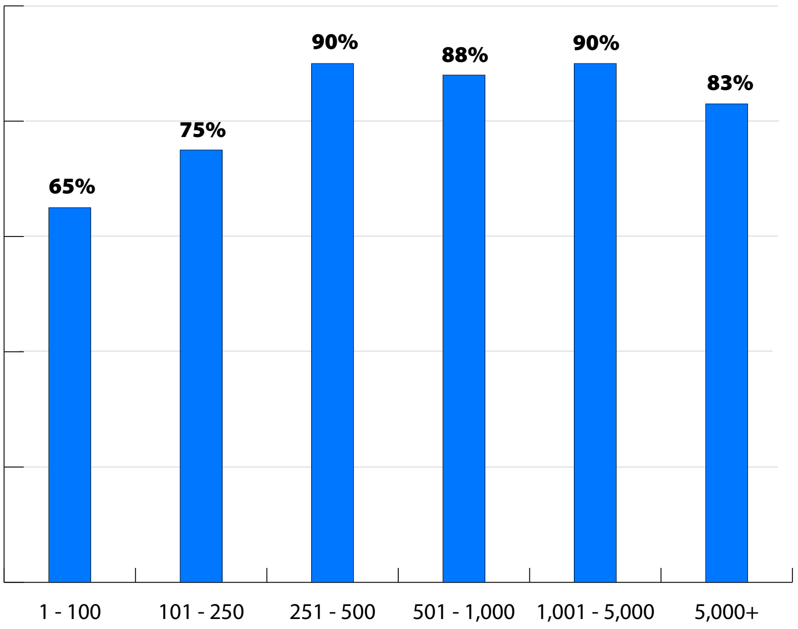
Despite these impressive numbers for Kubernetes, there are indeed situations in which Kubernetes is far more likely to be used. One such delimiting factor is organization size. When we compare the data on respondents' organization size given in the Demographics section with the data we gave above on Kubernetes usage rates, we find that bigger orgs are more likely to use Kubernetes. We found that 33% of organizations with 1-19 employees use K8s for container orchestration, while 64% of organizations sized 10,000+ use K8s. The below table shows how the usage rates of Kubernetes increases as organization size increases.

ORGANIZATION SIZE	KUBERNETES USAGE RATE
1-19	25%
20-99	21%
100-999	28%
1,000-9,999	36%
10,000+	32%

Figure 2

As we can see, there is a clear relationship between organization size and Kubernetes use. This same trend plays out when we compare Kubernetes use to containerization rates. In Figure 3, we have compared the data we collected by asking respondents how many containers their org runs in production with our data on K8s usage rates.

Figure 3: Number of Containers in Production

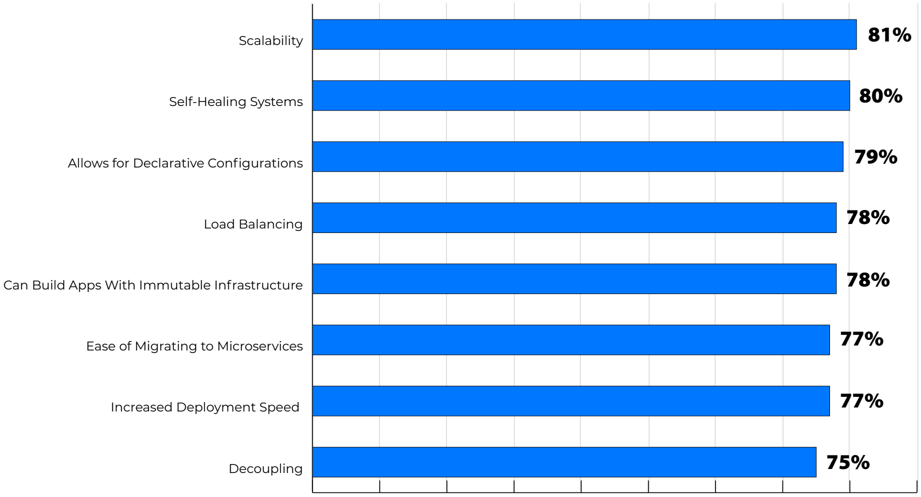


How and Why Kubernetes Is Used

When we look at the four main environments that make up the SDLC (development, QA/testing, staging, and production/deployment), we again find stable Kubernetes usage rates across the board. 74% of organizations who use containers in production/deployment use K8s, 73% of organizations who use containers in either QA or staging use K8s, and 72% of organizations who use containers in development also use K8s.

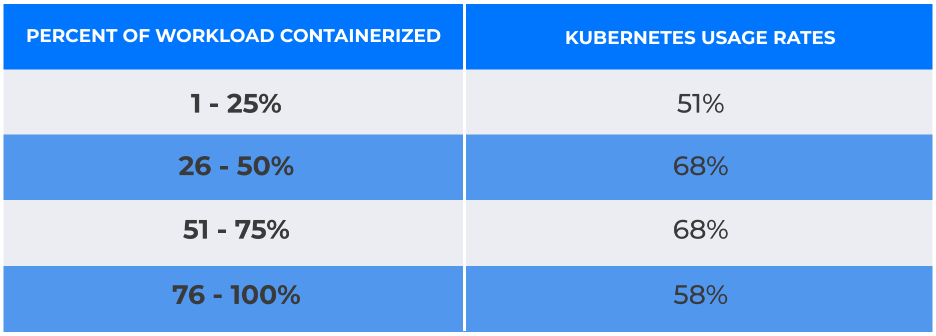
We also found that, despite organizations' reasons for adopting container orchestration tools, Kubernetes proved a popular solution to their needs. In Figure 4, we've compared the data we collected on the benefits of container orchestration tools with our data on K8s usage rates.

Figure 4: Reasons for Using Container Orchestration Tools



An organization's choice of PaaS solutions, interestingly, proved a significant factor in their decision to use Kubernetes. Among respondents whose organizations use OpenShift, 96% reported using K8s. This proved the most popular PaaS solution among Kubernetes users. Other popular PaaS solutions among Kubernetes users were Cloud Foundry (89%), Google App Engine (86%), and AWS Elastic Beanstalk (69%). It's interesting to note that, among these four tools, OpenShift, Google App Engine, and Cloud Foundry were all built with Kubernetes in mind and were originally built around open source projects.

Another variable that had an effect Kubernetes usage rates was the percent of an organization's workload that is containerized. Among respondents whose organizations have containerized 1-25% of their workload, 51% use K8s. For respondents whose organizations have containerized either 26-50% or 51-75% of their workload, 68% use K8s. And among organizations with 76-100% of their workloads containerized, 58% use K8s. Based on this analysis, it seems that organizations that containerized somewhere between 26-75% are the most likely to use container orchestration tools, and specifically use Kubernetes.



PERCENT OF WORKLOAD CONTAINERIZED	KUBERNETES USAGE RATES
1 - 25%	51%
26 - 50%	68%
51 - 75%	68%
76 - 100%	58%

Figure 5

Interestingly, while Kubernetes seems to be winning enterprise-level adherents across the software industry, developers are not yet quite as sold on it. Among respondents who told us that containers make their job easier, 68% use K8s. Among respondents who said that containers have made their job harder, however, 81% use K8s. Similarly, we found that 80% of respondents who claim that containers have had no impact on their job's difficulty use Kubernetes.

SPONSOR OPINION

Kubernetes Monitoring in Dynamic and Hybrid Environments

By Daniella Pontes, Senior Manager Product Marketing, InfluxData

Kubernetes, a.k.a. K8s, is paving the way to modern dynamic application environments. Its orchestration logic takes IT operations to the next level of automation of container clusters deployment, continuous updating and scaling. Visionaries have foreseen Kubernetes's ubiquitous adoption in enterprises and, most importantly, a critical contribution to the cloud-native transformation. Kubernetes is not only changing the way software is architected, integrated, and delivered to production environments, but also changing business models which now have the intrinsic potential of exponential growth and global distribution. One can say that Kubernetes is one of the most transformational technologies towards cloud-native today.

Applications fragmented into microservices are running on ephemeral containers and continuously integrated and delivered, and yet are running on hybrid environments, making monitoring Kubernetes for performance and reliability mandatory. In order to address this pressing need, Kuberenetes has integrated Prometheus monitoring model in its architecture.. However, Prometheus's endpoint 'pull monitoring' does only part of the job of collecting metrics. What security and implementation concerns arise when pulling data in multi-domain cloud environments? What happens when you want to monitor in events real-time, not at intervals? And what about applications that don't expose metrics in Prometheus format — that are better suited to other monitoring methods, such as pushing and streaming? Furthermore, what about monitoring various data types, numeric and non-numeric, with different retention policies (months, years... forever) and serving multiple customers and audiences, such as in managed services?

Push and pull metric collection mechanisms, stream ingestion, real-time analytics, high availability, and cost-effective long-term storage all matter when diving deeper into monitoring Kubernetes application environments of all sorts: cloud, hybrid, multi-cloud, and multi-IT. The reality is that most production environments don't have a singular approach to application deployment and monitoring. Therefore, one should consider solutions such as the InfluxDB time series platform that can handle variances, custom implementations, and unique business cases, while facilitating the need for evolution.

SPONSOR OPINION

InfluxDB

By Daniella Pontes, Senior Manager Product Marketing, InfluxData

Gravitational's Gravity offers multi-region automation for distributed applications. It combines a production-hardened deployment of Kubernetes with Teleport, Gravitational's multi-region SSH server, enabling clients to manage multiple deployments of Kubernetes applications across regions, data centers and cloud providers.

Multi-Region Kubernetes refers to when Operations teams in large companies with many distributed product teams need to provide Kubernetes-as-a-Service within their organization across multiple hosting regions and multiple hosting providers. Gravitational's search for a time series database suitable for configurable monitoring and alerting resulted in choosing InfluxData.

Using InfluxData for Kubernetes monitoring, Gravitational was able to make cloud applications portable for its clients and implement improvements that extended the power of Kubernetes and served their own need to scale the operational management of applications across many clusters.

- Transformation in their mindset, how they manage infrastructure and deliver products
- Full transparency throughout the environment enabling internal teams and customers to see how everything is running at any given time
- App metrics available the minute the app is "born" within one of their platforms

Category
Time Series Data Platform

Release Schedule
Quarterly Release cycles

Open Source
Yes

Strengths
- Built for developers
- Trusted by Ops
- Vital to business

Notible Users
- Capital One
- PayPal
- Comcast
- Wayfair
- Optum Health
- Gravitational

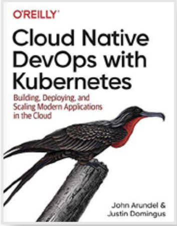
Website
influxdata.com

Blog
influxdata.com/blog

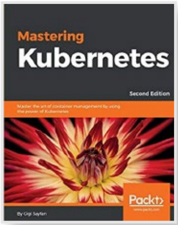
Twitter
twitter.com/influxdb

Diving Deeper Into Kubernetes in the Enterprise

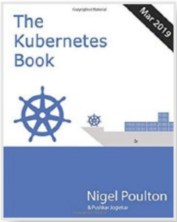
Books



Cloud Native DevOps With Kubernetes Learn how the Kubernetes ecosystem can help you create reliable applications with scalable infrastructure.



Mastering Kubernetes This book covers concepts like the advantages and disadvantages of running K8s on various cloud providers versus bare metal, monitoring and troubleshooting clusters, and customizing K8s.



The Kubernetes Book Get a thorough introduction to Kubernetes, starting from the basics. This book is one of the most popular guides to Kubernetes for beginners.

Refcards

Monitoring Kubernetes This Refcard outlines common challenges in monitoring Kubernetes, detailing the core components of the monitoring tool Prometheus.

Advanced Kubernetes This Refcard aims to deliver quickly accessible information for operators using any Kubernetes product.

Securing Your Kubernetes Deployment This Refcard will teach you the essentials of security in Kubernetes, addressing topics like container network access, user authorization, service token access, and more.

Zones

Cloud The Cloud Zone covers the host of providers and utilities that make cloud computing possible and push the limits (and savings) with which we can deploy, store, and host applications in a flexible, elastic manner. The Cloud Zone focuses on PaaS, infrastructures, containerization, security, scalability, and hosting servers.

Open Source The Open Source Zone offers practical advice regarding transitioning from a closed to an open project, creating an enforceable code of conduct, and making your first OSS contributions. This Zone encourages you to adopt an open-source mentality and shape the way open collaboration works.

DevOps DevOps is a cultural movement, supported by exciting new tools, that is aimed at encouraging close cooperation within cross-disciplinary teams of developers and IT operations. The DevOps Zone is your hot spot for news and resources about continuous delivery, Puppet, Chef, Jenkins, and much more.

Podcasts

PodCTL Produced by Red Hat OpenShift, this podcast covers everything related to enterprise Kubernetes and OpenShift, from in-depth discussions on Operators to conference recaps.

Deloitte on Cloud This episode of the Deloitte on Cloud podcast dives into a few ways that organizations can use Kubernetes to standardize processes around cloud migration.

Kubernetes Podcast from Google Considering that Google produces it (and that Google also created Kubernetes in 2014), you might call this podcast a classic. Enjoy weekly interviews with prominent tech folks who work with K8s.

ORIGINAL RESEARCH

Executive Insights on the State of K8s in the Enterprise

By Tom Smith, Research Analyst at DZone

To understand the current and future state of Kubernetes (K8s) in the enterprise, we gathered insights from IT executives from 22 companies. Here's what we learned:

1. Security, planning, people with skills in Kubernetes (K8s), and data locality are four of the keys mentioned most frequently for the successful adoption and implementation of K8s. Think about how to claim and reclaim storage resources to deal with security, performance, reliability, and availability — all of the traditional data center operations concerns. You should have the same concerns for K8s as when you put software into production — security, monitoring, and debugging.

Build your environment to be specific to a purpose, not to a location. Have a plan driven by your goals. Start with people that have knowledge of K8s that will work well together when services are divided among teams. The team needs to know what's going on across the landscape as well as to understand what's required for "day two" operations — upgrades, patches, disaster recovery, and scale.

TREND PREDICTIONS

- Security, planning, knowledge, and data locality are four keys to success with Kubernetes in the enterprise.
- Kubernetes will continue to make it easier to scale and achieve speed to market in a vendor-agnostic way.
- The most common failures with Kubernetes deployments are around the lack of skills/knowledge, complexity, security, and "day two" operations.

Think about how to handle state, whether it's using stateful sets leveraging your provider's block storage devices, or moving to a completely managed storage solution, implementing stateful services correctly the first time around is going to save you huge headaches.

2. Kubernetes has made it easier to scale and achieve speed to market in a vendor-agnostic way. We're seeing deployments in production at scale with thousands, tens, and hundreds of thousands of containers implementing microservices. K8s has provided the infrastructure that's more stateless, self-healing, and flexible. K8s enables teams to scale production workloads and fault tolerance not previously possible.

K8s is faster to scale and deploy, more reliable, and offers more options. It lets both the application and platform teams move more quickly. Application teams don't need to know all the details, and platform teams are free to change them.

A single configuration can be propagated across all clusters. K8s features ubiquitous platform and cloud provider support. K8s provides a completely uniform way of describing containers and all other resources needed to run them: databases, networks, storage, configuration, secrets, and even those that are custom-defined.

3. K8s enhances the security of containers via roll-back access control (RBAC), reducing exposure, automation, and network firewall policies. Regarding security, K8s solves more problems than it creates. RBAC enforces relationships between resources like pod security policies to control the level of access that pods have to each other. K8s provides the access and mechanisms to use other things to secure containers.

The security benefits of containers and K8s outweigh the risks because containers tend to be much smaller than a VM running in NGINX which will have a full operating system with many processes and servers. Containers have far less exposure and fewer attack surfaces.

By automating concepts and constructs of where things go using rules and a stabilized environment, you eliminate a lot of human error that tends to occur in a manual configuration process. K8s standardizes container deployment — set it once and forget it.

Due to the increased autonomy of microservices deployed as pods in K8s, it's important to have a thorough vulnerability assessment on each service, and to change control enforcement on the security architecture. A strict security enforcement is critical to defend against security threats. It's important to attend to things like automated monitoring/auditing/alerting, OS hardening, and continuous system patching.

> Due to the increased autonomy of microservices deployed as pods in K8s, it's important to have a thorough vulnerability assessment on each service.

4. K8s use cases tend to be related to speed, CI/CD pipelines, cost reduction, legacy modernization, and scale. Building on top of K8s has helped smaller teams move faster. It accelerates every phase of the application lifecycle and reduces time to market. It helps automate DevOps tasks and builds in best practices quickly. K8s makes it possible to adopt continuous development and deployment by ensuring deliveries are made to the right place at the right time.

Better performance results in more cost savings, and K8s helps reduce infrastructure costs. K8s also helps reduce technical debt as organizations pursue legacy containerization/modernization. There's also automatic scale-in and -out to adjust quickly to application workload demands, and to maintain integrity when scaling.

5. The most common failures revolve around a general shortage of skills/knowledge, as well as around complexity, security, and "day two" operations. K8s talent is very hard to find and retain. There is also a lack of understanding about how K8s functions. A common challenge is how to get a team up to speed quickly. Recruit experts with the depth of knowledge of K8s and DevOps required to build proper tools and implement application workflows in containerized environments.

People often give up on implementation because it's too hard. You can expect significant complexity and a steep learning curve. People underestimate the complexity of installing and operating K8s. It's easy getting started, but then people are surprised by the complexity when they put it into production with security and monitoring in place.

Enterprises can find it challenging to implement effective security solutions. We see security and operations failures arise when teams don't implement any policy around the creation of external load balancers/ingresses. We see failures around security, with new vulnerabilities weekly that require patches. "Day two" operations like upgrades and patches need to be managed. There's an ongoing need to deploy persistent storage, to monitor and alert on failure events, and to deploy applications across multiple K8s clusters.

6. Concerns regarding the current state of K8s also center on complexity, security, and finding people with sufficient skills. Complexity is a big issue. Deploying K8s is a relatively new practice, and it can be very challenging to pick the right combination of technologies and tools.

Security controls lag behind, and newcomers may adopt inadequate K8s security measures — allowing attackers with increasingly sophisticated exploits to succeed. You cannot assume that managed K8s offerings are somehow inherently secure — or that by limiting CI/CD access to just a few DevOps people, that any risk can be avoided.

People who try to implement K8s on their own have trouble maintaining their own platform. Organizations also tend to assume they need K8s when they don't. A lot of people are flying blind, running random containers with third parties without monitoring them. People assume it's self-healing, and ignore the details.

7. Driven by adoption of the cloud and IoT, K8s is destined to become the de facto platform for developers. It will become the standard platform for running applications, similar to the initial excitement in the developer community around Java. The future is in IoT, with K8s enabling communication and rollbacks. You'll be able to make IoT device nodes in a larger K8s cluster for faster updates and more services. The K8s cloud operating system will also extend to hybrid, multi-cloud operating systems.

There will be more externalization of the platform and enterprise hardening of K8s. It will become more stable at the core while also becoming more extensible. The toolkit for K8s operators will capture more complicated lifestyle automation. Containers will eventually replace virtual machines and will support other infrastructure further up the stack. The technology will be increasingly standardized, stable, and portable going forward. The adoption of open-source strategy and K8s by businesses will continue to rapidly grow, with an ecosystem backed by leading internet tech companies and an expanding K8s developer community.

8. When working with K8s, developers need to keep in mind security, architecture, and DevOps methodology. Developers and DevOps engineers need to consider how best to secure their K8s environments. Developers should become familiar with the Cloud-Native Computing Foundation (CNCF) stack, with K8s as the centerpiece — along with technologies like network meshes, permit use, and runtime security.

Figure out the best architecture to build around, but architect so your application can run on a different platform. Understand the need to be elastic on a cloud-native platform, but be explicit about the shape of your infrastructure and be explicit with your manifests.

Figure out the best architecture to build around, but architect so your application can run on a different platform.

Leave the large scale to Ops teams or an external company and focus on building cloud-native applications using cloud-native principles. Try to get DevOps automation done right — have a good CI/CD process and make it as repeatable as possible. Use agile techniques. Do incremental development for fast feedback and use best-of-breed tools.

Finally, make the trip to KubeCon. If you experience a problem, reach out to the community.

Below is the list of executives who were kind enough to share their insights with us:

- Dipti Borkar, VP of Product Management and Marketing, Alluxio
- Matthew Barlocker, Founder and CEO, Blue Matador
- Carmine Rimi, Product Manager Kubernetes, Kubeflow, Canonical
- Phil Dougherty, Sr. Product Manager, DigitalOcean
- Tobi Knaup, Co-founder and CTO, D2iQ

- Tamas Cser, Founder and CEO, Functionize
- Kaushik Mysur, Director of Product Management, Instaclustr
- Niraj Tolia, CEO, Kasten
- Marco Palladino, CTO and Co-founder, Kong
- Daniel Spoonhower, Co-founder and CTO, LightStep
- Matt Creager, Co-founder, Manifold
- Ingo Fuchs, Chief Technologist, Cloud and DevOps, NetApp
- Glen Kosaka, VP of Product Management, NeuVector
- Joe Leslie, Senior Product Manager, NuoDB
- Tyler Duzan, Product Manager, Percona
- Kamesh Pemmaraju, Head of Product Marketing, Platform9
- Anurag Goel, Founder and CEO, Render
- Dave McAlister, Community Manager and Evangelist, Scalyr
- Idit Levine, Founder and CIO, Solo.io
- Edmond Cullen, Practice Principal Architect, SPR
- Tim Hinrichs, Co-founder and CTO, Styra
- Loris Degioanni, Founder and CTO, Sysdig