RAPID7

WHITEPAPER

Cloud & Kubernetes - Removing the Security Blind Spots

Best Practices for Visibility, Control, Security, and Compliance of Kubernetes Deployments

TABLE OF CONTENTS

Executive Summary

Cloud Security Posture Management (CSPM) and Cloud Workload Protection Platforms (CWPP) are foundations for good cloud security, but with more and more organizations adopting microservices and Kubernetes orchestration using cloud and hybrid cloud infrastructure, they are unwittingly expanding their significant attack surface. As engineering teams migrate existing applications to the cloud or build new cloud-native applications, security teams face the considerable task of protecting these deployments in a relatively new, foreign, and often hostile environment. At the heart of these application deployments is Kubernetes.

The percentage of organizations using Kubernetes skyrocketed to nearly 50% in 2020, up almost two-fold since 2018. As a new application delivery vehicle, designed from the ground up to be cloud native, Kubernetes requires a new security approach that builds on existing cloud infrastructure security practices, yet respects the sheer power, flexibility, and ease of deployment that Kubernetes offers.

Kubernetes is a cloud unto itself, requiring an integrated and holistic security implementation. Cloud security solutions, such as CSPM and CWPPs, provide aspects of Kubernetes coverage. However, adoption of these solutions is still early. Without an understanding of why Kubernetes requires a new security approach, there is a significant risk of leaving blind spots that obscure visibility and create control gaps.

If security teams do not prioritize and close these gaps, the entire cloud application stack, and the organization as a whole, are at risk. However, if security teams implement the necessary hardening, visibility tools, and guardrails to eliminate the blind spots, they will not only manage these risks but also fuel innovation by empowering the flexibility, dynamism, and speed of Kubernetes.

In this paper, we explore:

- Why Kubernetes security can leave blind spots and why eliminating them is not as simple as implementing solutions that have become the standard for cloud security – CSPM and CWPP.
- Why deep visibility and control are the linchpins of protecting modern cloud infrastructure, inclusive of Kubernetes.
- Why comprehensive Kubernetes security requires holistic security and posture management for the entire cloud stack, including configuration management, identity and access management (IAM), infrastructure as code (IaC), and compliance management.

This is the new imperative: cloud security teams must empower Kubernetes.

The Need for Full Cloud Stack Visibility and Control

The Rise of Kubernetes

Security teams face the considerable task of managing the confidentiality, integrity, availability, and compliance of cloud application deployments. Their DevOps teams are pushing applications to the cloud with gusto, leveraging a host of server-based and serverless cloud services including infrastructure, containers, platforms, software, and functions as a service (IaaS, CaaS, PaaS, SaaS, and FaaS).

At the heart of the cloud application stack is the workload orchestration layer, typically running Kubernetes. Kubernetes orchestrates container deployments, though, increasingly, it is orchestrating workloads on top of public cloud infrastructure.

Kubernetes use has skyrocketed significantly, with nearly half of all organizations using it in 2020, up from 27% in 2018. Why? Enterprises report clear benefits from adopting Kubernetes, most notably better resource utilization, shortened software development cycles, containerizing monolithic applications, and enabling a move to the cloud, according to VMware's "State of Kubernetes 2020 Report."[1] In other words, Kubernetes drives innovation.

This rapid adoption places Kubernetes front and center on the security team's radar. The team's challenge is that Kubernetes plays a central role in the cloud application stack. Its inherent characteristics of dynamism and complexity pose a significant and potentially catastrophic risk to the organization. If security teams cannot manage Kubernetes' security and compliance posture and ensure the security of the workloads it orchestrates, the entire cloud application stack will be at high risk. If teams can mitigate this risk, they will empower Kubernetes and the innovation it drives.

Kubernetes Security and Compliance Leaves Blind Spots

Teams looking for solutions to protect cloud applications have many offerings to choose from (e.g., Cloud Access Security Brokers (CASB), Cloud Security Posture Management (CSPM), Cloud Workload Protection Platforms (CWPP), and Cloud Infrastructure Entitlement Management CIEM). The most mature of these are CSPM and CWPP. Cloud workload protection refers to the unique requirements of safeguarding workloads running in a cloud, multi-cloud, or hybrid cloud architecture. A CSPM solution protects the infrastructure on which the workloads run.

[1] https://k8s.vmware.com/state-of-kubernetes-2020/

Kubernetes can be thought of as a cloud unto itself, given its on-demand, declarative nature, robust scalability, and platform independence. Therefore, the security and compliance of a Kubernetes deployment require aspects of multiple cloud security offerings. But, as with most security controls, just slapping together a few different solutions (e.g., CSPM and CWPP) does not guarantee Kubernetes security. Worse, this approach could lead to a false sense of security due to significant potential gaps in Kubernetes security and compliance coverage.

The best way for security teams to protect Kubernetes is to take a step back from the CSPM vs. CWPP vs. CXYZ decision. It is best to first focus on Kubernetes' central role in cloud application workload deployment and its relationship to the underlying cloud infrastructure. Without this understanding, organizations are at risk of developing significant blind spots due to missing coverage and control.

Kubernetes Orchestration Is Great Until It's Not

Applications are workloads that run on cloud infrastructure (e.g., VMs and storage repositories), containers, or serverless compute services – including functions as a service (FaaS) – and containers as a service (CaaS). For a workload running on a container, Kubernetes is the orchestrator, decoupling the workload from the underlying container infrastructure.

As an orchestration service, Kubernetes is declarative, continually managing any deltas between the desired state (e.g., six active containers) and the actual state (e.g., five containers responding). When a container hangs or fails, Kubernetes automatically spins up another container to close the gap between the real and desired states.

This agility and automation are a boon for application deployment. However, if not instrumented correctly, Kubernetes dynamics is a nightmare for the security team. For example, that failed container might be an in-scope (i.e., Payment Card Industry (PCI)-compliant) workload in a cardholder data environment (CDE). What if Kubernetes automatically restarts that workload on a container outside the CDE? Considering a cloud application may consist of thousands of containers, and the average life of a container is less than five minutes, the chances of this happening are high.[2]

When the security team does not have complete visibility into and control over the Kubernetes orchestration layer, the dynamics of a container environment (e.g., the wayward workload) significantly increases the organization's cloud application risk.

So, what level of visibility and control are necessary to make sure Kubernetes orchestration falls within the boundaries of good security and compliance practices? Answering this question requires diving a bit deeper into Kubernetes.

[2] https://www.zdnet.com/article/technology-containers-short-lifespans-are-getting-even-shorter/

A Need to Fly Both Planes: Control and Data

In cloud infrastructure, each cloud service has a control plane and a data plane. The control plane governs the infrastructure and the data plane supports the workloads. As shown in Figure 1, a Kubernetes cluster also contains both a control plane (e.g., API server, controller manager, etcd) and a data plane (e.g., kubelet, kube-proxy).

And here lies the reason why Kubernetes security can result in blind spots. Many security teams are trying to protect Kubernetes deployments via one plane or the other. For example, some rely on the Kubernetes cluster metadata to monitor the application but have little insight into what is happening to the workload running on a container, inside a pod, inside the cluster. Others are tracking the data plane and missing compromise of the Kubernetes Master, unauthorized access, modification of etcd, or other significant control plane violations.

Securing a Kubernetes deployment requires deep visibility into and control over both the Kubernetes data and control planes.

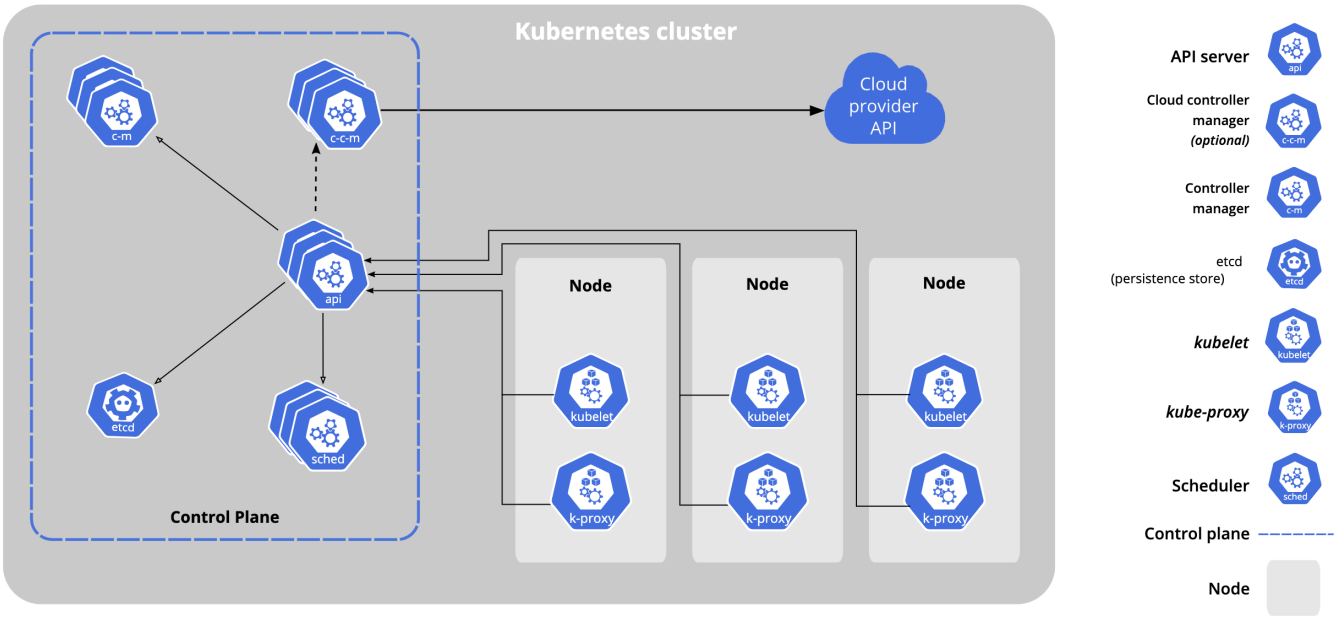

Figure 1: Kubernetes Components

The Need for Full Cloud Stack Posture Management and Workload Protection

Achieving the necessary deep visibility into and control of both planes requires aspects of both posture management and workload protection. In other words, ensuring security and compliance of Kubernetes deployments requires the monitoring, configuration management, and guardrails of CSPM and the real-time, deep visibility and granular control of CWPP. Plus, because Kubernetes sits in the middle of the cloud application stack, visibility and control must be consistent across the entire stack. For example, locking down Kubernetes identity and access management (IAM) without aligning with the underlying cloud infrastructure IAM (e.g., IaaS, PaaS, FaaS) can leave a significant gap in coverage due to mismatched privileges and identities.

Three Steps to Kubernetes Deployment Security and Compliance

Once you have a better understanding of the underlying dynamics and architecture of Kubernetes and its central role in the cloud application stack, achieving the necessary Kubernetes posture management and workload protection is a three-step process:

1. **Hardening.** Instrument and harden (using? best practice configuration settings) the Kubernetes orchestration layer.

2. **Visibility and control**. Implement deep visibility and control to detect immediately – and mitigate –any missing or changed settings during runtime. This step also includes detecting anomalous and malicious activity.

3. **Runtime guardrails**. Implement guardrails that define the range of settings and configurations meeting the cloud application's security and compliance requirements during runtime. Security needs include tight governance of core cloud functions such as identity, infrastructure as code, and compliance.

Instrumenting and Hardening the Kubernetes Environment

The first step is instrumenting and hardening Kubernetes for a secure and compliant deployment. Though this is a complex process, there are four primary actions that an organization must take:

1. Configure and use Kubernetes-native security controls. For example, native to Kubernetes are role-based access control (RBAC), pod security policies (PSP), network policies, and secrets management. One config change during deployment (e.g., exposing an RDP port) could lead to a severe breach.

2. Harden both the Kubernetes control and data plane configurations. There are heaps of settings that DevOps and security must set correctly to harden a Kubernetes deployment (e.g., Seccomp, AppArmor).

3. Augment Kubernetes-native functions with additional controls, including micro-segmentation firewalls, encryption, and image scanning.

4. Configure and protect the broader Kubernetes ecosystem (e.g., using Service Mesh).

The details of implementing these capabilities go beyond the scope of this paper. However, the Cloud Native Computing Foundation (CNCF) provides one of the best architectures for Kubernetes security.[3] As shown in Figure 2, the workload orchestration security model covers native security functions of Kubernetes and the additional controls necessary for instrumenting and locking down Kubernetes.

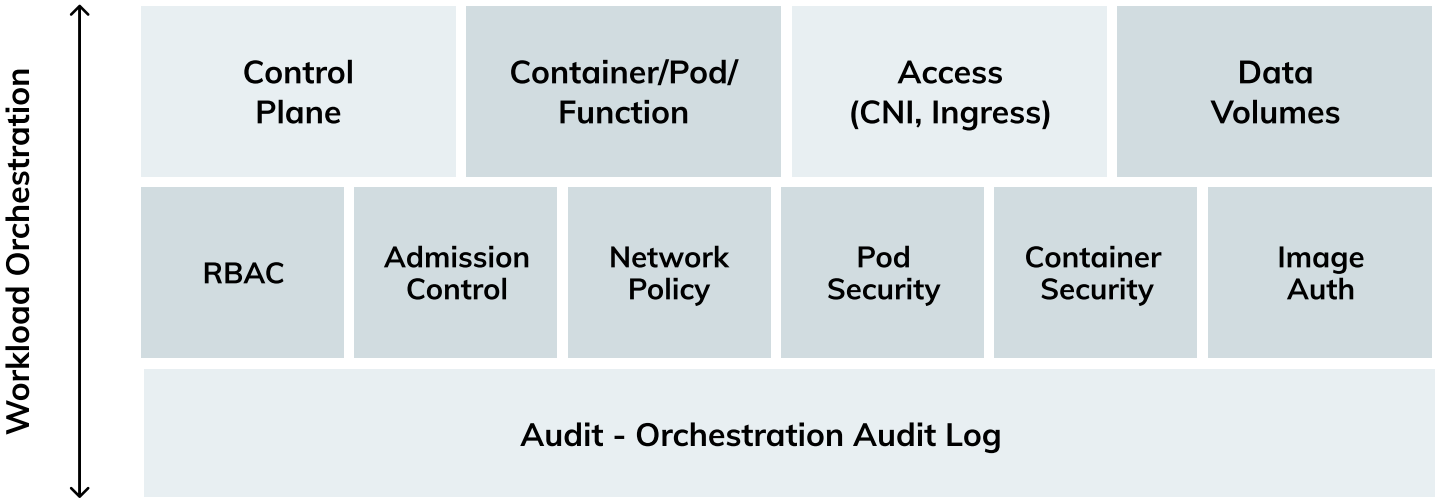

Figure 2: CNCF Workload Orchestration Security Model

Implementing Deep Visibility and Control for Kubernetes

After locking down Kubernetes, the next step is establishing deep visibility to achieve and maintain the security and compliance of a Kubernetes deployment.

This visibility is essential because once a container spins up, it should be immutable (no changes to its configuration). A runtime change to a container could be indicative of a Kubernetes breach. Therefore, any changes to a running container must trigger immediate action. As the CNCF states, “It is important to monitor and detect any changes to the initial configurations made in runtime to ensure the continued security posture of the cluster.”

Delivering this level of visibility requires deep hooks into the Kubernetes/container environment. For example, it is imperative to see core Kubernetes attributes, such as pod name, type, deployment, namespace data, user access, container start-stop, and container image status.

Obtaining this level of Kubernetes visibility is not possible through audit log monitoring alone. Additionally, the security approach must provide kernel-level visibility into all Kubernetes activity, configuration settings, and security controls. This visibility opens the door to monitoring and enforcing organization controls based on pre-defined clusters' profile and compliance requirements.

[3] Source: CNCF Cloud-native Security Whitepaper

Detecting Anomalous Kubernetes Behaviors and Threat Actors

Going deeper, detecting configuration changes is important but not sufficient. The Kubernetes API requires monitoring with forensics to see if these configuration changes are anomalous behavior. This requirement is essential within multi-cluster Kubernetes environments to prevent sensitive workloads from moving across cluster borders (e.g., our wayward PCI payload).

Also, security must account for threat actors. After all, one of the best ways to differentiate a benign anomaly from a malicious anomaly is to align with the threat landscape. This alignment requires fine-grained detection at the container/pod level that correlates with threat feeds to immediately identify indicators of compromise (IoC) and potentially even identify tactics, techniques, and procedures (TTPs).

Detection without Response Is a Non-Starter

Finally, simply detecting a Kubernetes configuration change or anomalous activity and then alerting a security information and event management (SIEM) or security orchestration automation and response (SOAR) system is a non-starter for protecting a Kubernetes deployment. Things move too fast to control changes or anomalies manually and indirectly. With containers, workloads come and go in seconds.

The security team must institute detection and response rules that trigger automated or semi-automated responses to detected configuration changes and anomalous behavior. However, given Kubernetes' central role in the application stack, this automation cannot operate in a vacuum. Any automated response for Kubernetes must be in the context of full cloud stack posture management.

Holistic Kubernetes Security and Posture Management

The last step for Kubernetes security and compliance is implementing guardrails to protect and govern Kubernetes from a full-stack perspective. These guardrails are essential for preventing security drift by enforcing security best practices for application deployments, thus securely enabling Kubernetes' speed of innovation. These security best practices include implementing benchmarks for Kubernetes configurations and least privilege access (LPA) with cloud IAM, governing infrastructure as code (IaC) templates, and maintaining continuous regulatory and Industry compliance.

Configuration Guardrails

Kubernetes has hundreds of possible configuration settings. Many of these settings have profound security and compliance implications. To address this, the Center for Internet Security (CIS) has released a series of benchmarks or prescriptive security guidelines for configuring both the Kubernetes control and data planes. Protecting a Kubernetes deployment requires running these benchmarks automatically when deploying and then tracking in real time when a configuration change differs from a benchmark recommendation. The CIS benchmarks become a guardrail that permits flexibility to adjust a Kubernetes deployment during runtime without allowing changes to negatively affect the cluster's security and compliance posture.

Cloud IAM

Essential to protecting a Kubernetes deployment are establishing and managing LPA. To do this requires IAM guardrails throughout the cloud application stack, including Kubernetes. At a minimum, Kubernetes must be wired with controls to detect, monitor, and act on any RBAC and admission controller activity. This IAM control must align with the broader cloud stack IAM. For example, an application running on Amazon EKS must authenticate to access other APIs for many different functions running outside EKS (e.g., compute, storage, database, machine learning).

Cloud IaC

As a declarative infrastructure, a Kubernetes deployment is heavily reliant on IaC. Infrastructure as code uses a configuration language (e.g., Terraform) that defines what the infrastructure (e.g., a Kubernetes cluster) should look like at the end state, rather than prescribing the steps necessary to get there.

On the plus side, there are benefits to using IaC with Kubernetes, including reducing human error, maintaining consistency, improving change tracking and auditing, and accelerating recovery in a catastrophic failure.

On the downside, because Kubernetes has a control plane and interacts with workloads throughout the cloud stack, organizations require a solution that scans IaC and API calls for the entire cloud stack, including Kubernetes clusters.

Cloud Compliance

Finally, an essential Kubernetes guardrail is maintaining regulatory and industry compliance. The intentional dynamism of Kubernetes results in containers' continually spinning up, spinning down, and moving around. Any one of these actions could violate an overarching compliance requirement such as PCI, General Data Protection Regulation (GDPR), or Health Insurance Portability and Accountability Act (HIPAA)). Plus, compromise of the Kubernetes data plane can easily lead to a catastrophic data breach.

Maintaining compliance in a Kubernetes deployment requires implementing policy templates and continual cloud-specific implementation (e.g., EKS, GKE, AKS) checks, similar to running CIS benchmarks. Running these checks mandates deep visibility into the Kubernetes environment, the infrastructure on which it is running, and the workloads it is supporting.

Conclusion

Following the three-step process to protect Kubernetes deployments removes potential Kubernetes security blind spots by implementing both workload and posture management. This approach also gives security and compliance teams the flexibility to select the right cloud security solution by focusing on the functional requirements that best meet the organization's needs. These functional requirements include:

- Kubernetes hardening
- Deep visibility and control of data and control planes, including detection of anomalous behavior and threats
- Guardrails to protect the cluster configuration, IAM, IaC, and compliance of the entire cloud application stack deployment

Though Kubernetes security is challenging, security teams can manage risk effectively by taking this approach to secure Kubernetes deployments. Plus, with this approach, teams will free Application Development and DevOps teams to better innovate by securely enabling the flexibility, dynamism, and speed of Kubernetes.

Request a personalized demo.