

Whitepaper

A How-To Guide for Hardening Kubernetes Security with Certificate Lifecycle Management

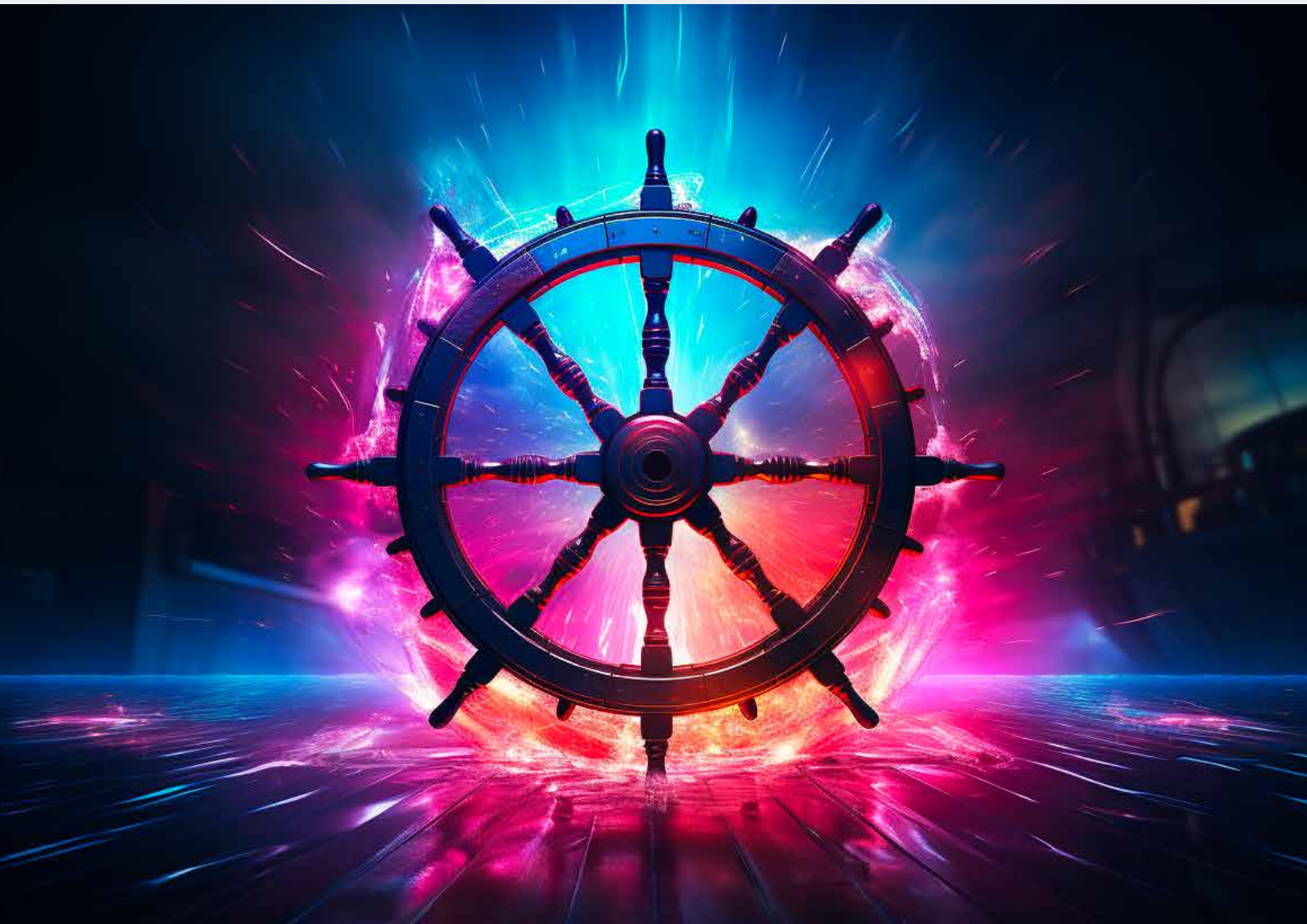


Table of Contents

01. As Kubernetes Adoption Soars, So Does the Need for Robust, Developer-Friendly, and Cloud-Native Security Solutions.....	03
02. Kubernetes Security: Complexities, Challenges, and Risks.....	06
03. Kubernetes Security Incidents Are Growing at an Alarming Rate	09
04. The Way Forward – Implement Strong Identity Protection	11
05. But...Issuing Trusted Identities Is Not Enough, Managing Them Effectively Is Key.....	15
06. How AppViewX KUBE+ Simplifies Certificate Lifecycle Management in Kubernetes.....	20
07. Keep Your DevOps Speed Up and Kubernetes Environments Secure with Identity Protection and Effective Certificate Lifecycle Management	22

01



As Kubernetes Adoption Soars, So Does the Need for Robust, Developer-Friendly, and Cloud-Native Security Solutions

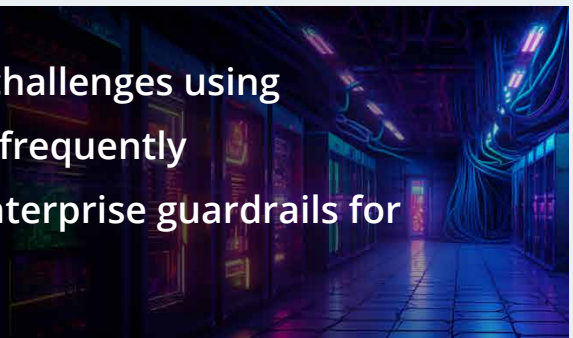
You are not moving fast enough if you are not on the cloud is the prevailing wisdom shaping the digital transformation era.

A staggering **94% of enterprises today** use cloud services and the number continues to grow everyday. Cloud-native technologies, such as microservices, containers, and Kubernetes, are key catalysts driving the adoption, transforming how applications are developed, managed, delivered, and consumed.

It comes as no surprise that over **78% of production workloads** today are deployed as containers or serverless applications. With increased container usage, Kubernetes, the container orchestration platform, is also seeing unprecedented adoption.

Thanks to its inherent benefits, such as built-in deployment, scalability, and resiliency, Kubernetes has become the go-to platform for managing containerized applications.

As the adoption of Kubernetes rises, security teams are growing increasingly worried about various security challenges and risks they bring. A vast array of components and the rapid use of open-source software make Kubernetes environments highly susceptible to injection flaws, API attacks, and malicious lateral movement that can turn into large-scale supply chain attacks.



“98% (of surveyed enterprises) face challenges using Kubernetes in production. The most frequently experienced challenge is ensuring enterprise guardrails for Kubernetes environments.”

[The New Frontiers of Kubernetes, 2023 State of Production Kubernetes](#)

Relying solely on the default security controls and configurations offered by cloud service providers (CSPs) and Kubernetes can be risky as these are typically limited in functionality and only cover some security aspects. Protecting multiple layers of the complex Kubernetes stack requires a more comprehensive security approach.

However, securing cloud-native Kubernetes deployments has proven to be a lot more challenging than securing traditional data centers. Conventional security solutions, built to guard the perimeter, are not sufficient to protect Kubernetes environments that are highly dynamic, distributed, and scalable. Extending perimeter-based controls to Kubernetes has only added to the complexity, creating security blind spots and weak links.

To navigate these operational complexities and risks, there is a growing demand for cloud-native security solutions that are developer-friendly, offer visibility into workloads and applications, help prevent threats proactively, and ensure continuous compliance - all without impeding the speed and agility of DevOps.

While several approaches help address Kubernetes security concerns, identity is at the center of it. Provisioning every “thing” or machine (e.g., workloads, services, code) interacting with the Kubernetes environment with a trusted identity and authenticating connections helps control access to Kubernetes clusters, effectively minimizing security risks. A key enabler of this process is machine identity management, powered by public key infrastructure (PKI) and certificate lifecycle management.

This whitepaper delves into the challenges and risks associated with securing Kubernetes environments and explores how a robust identity-first strategy and a reliable certificate lifecycle management solution can help overcome these challenges in hardening Kubernetes security.

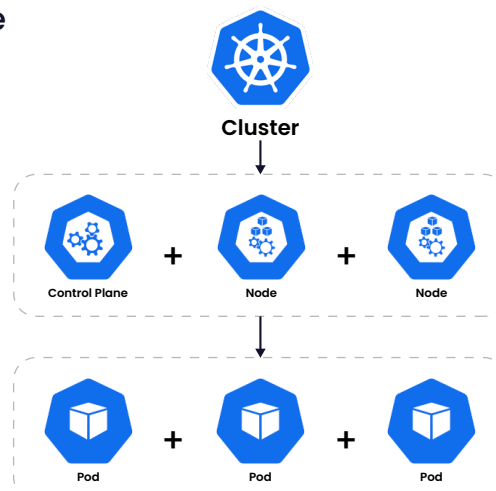


02

Kubernetes Security: Complexities, Challenges, and Risks

As many would agree, Kubernetes is a very complex technology made up of a large number of moving parts. It also supports an environment that is highly dynamic in nature. Imagine a Kubernetes environment in production - it typically includes several clusters hosted across multiple kinds of infrastructure (public cloud, private cloud, on-prem, edge), using multiple distributions (cloud-managed, self-hosted, edge-specific), with a variety of software elements (open-source and commercial). Each of these adds an additional layer of complexity, making the overall environment extremely demanding to both manage and secure.

Kubernetes Architecture



Further, technology teams expect to do more with Kubernetes in the coming year by growing the number of container-based applications built for Kubernetes as well as migrating existing applications to Kubernetes.

Navigating the inherent complexities of the Kubernetes environment and overseeing an expanding portfolio of cloud-native applications has made life hard for DevOps, PlatformOps, CloudOps, and Security teams. Despite Kubernetes offering basic built-in security features, security teams need help taking control of the environment, necessitating additional security solutions that are capable of protecting all layers of the Kubernetes infrastructure without causing friction.

Security Challenges and Risks in Kubernetes

- **Lack of Visibility**

Cloud-native applications today use a large number of (dozens to hundreds) containers that are constantly spun up or down, created, deleted, or rescheduled to meet fluctuating demand. Due to the ephemeral lifespan of containers, maintaining visibility of all containers across a fleet of clusters and multiple cloud environments becomes highly challenging.

- **Compromised Containers**

Containers and pods need to talk to each other within the cluster and to external services to accomplish their tasks. In such an interconnected setup, even if a single container is breached, the impact is multifold as the attacker can quickly move from the compromised pod to all the other pods and containers connected to it. By default, resources in Kubernetes are not isolated. Hence, lateral movement or privilege escalation is not forbidden, maximizing the impact of a successful compromise.

- **Control Plane Exposure**

The control plane is the brain of Kubernetes, with several critical components that work together to ensure that the cluster maintains the desired state and that workloads are running as expected. Because of these powerful capabilities, the control plane is a prime target for attackers. At the same time, lack of network separation and authentication exposes control plane components to the internet or an untrusted network, making it easy for attackers to gain unauthorized access and compromise

- **Misconfigurations**

The etcd database is the most important element within the control plane that stores Secrets - sensitive cluster information, such as credentials, OAuth tokens, and SSH keys. Gaining etcd access is equal to gaining root access to the cluster. So, the database must be configured to only trust and communicate with the Kubernetes API server.

However, misconfigurations have become highly prevalent, allowing attackers to exploit insecure ports and gain API access to etcd. Also, Secrets are often mishandled by developers who store them in YAML files, environment variables, or directly within container images. As Kubernetes does not encrypt Secrets at rest by default, they can be easily exfiltrated by attackers.

- **Disjointed Processes and Team Silos**

As mentioned earlier, organizations today run a large number of Kubernetes clusters, primarily managed by DevOps and CloudOps teams. While DevOps focuses on getting work done quickly, CloudOps focuses on platform availability. Between these priorities, security takes a backseat.

The problem is further compounded by the use of disparate tools, resulting in delays, complexity, and inconsistency issues. On the other hand, Security teams have little visibility or control over DevOps practices, leading to blind spots, vulnerabilities, and compliance violations.

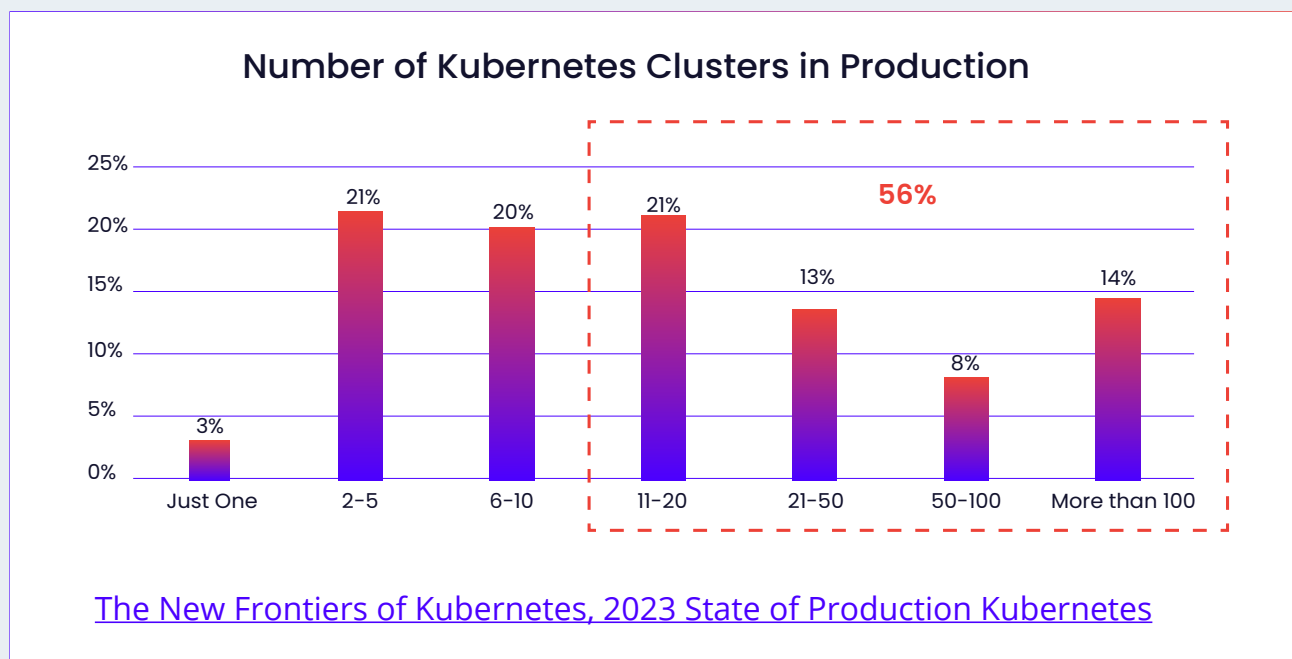
03

Kubernetes Security Incidents Are Growing at an Alarming Rate

It is almost impossible to overlook the growing interest of threat actors in containers and Kubernetes environments. According to [the Red Hat State of Kubernetes Security Report, 2023](#), 90% of surveyed respondents experienced at least one security incident in their Kubernetes environments in the last 12 months.

During [a three-month-long investigation, the research team at Aqua Security](#) discovered widespread attacks on Kubernetes clusters across hundreds of organizations. Alarming findings revealed that clusters belonging to more than 350 organizations, open-source projects, and individuals were detected as openly accessible and unprotected. A notable subset of these clusters was connected to large conglomerates and Fortune 500 companies. What's more alarming is that at least 60% of these clusters were breached and had an active campaign with deployed malware and backdoors.

The root cause of these attacks lay in misconfigurations that either allowed anonymous access with privileges or unknowingly exposed the Kubernetes cluster to the internet.



What's concerning is not just the scope of attacks, but also the speed at which Kubernetes clusters become targets. An experiment conducted by the [Wiz Threat Research Team](#) found that it takes only 22 minutes for a newly created Kubernetes cluster to start receiving initial malicious scanning attempts.

These revelations underscore a clear reality - Kubernetes stands out as a prime target for attackers. As the adoption of Kubernetes is projected to soar in the coming year, there is an immediate need to implement robust and tailored security measures that can effectively fortify the environment without compromising its performance.

"In the wrong hands, access to a company's Kubernetes cluster could be business ending. Proprietary code, intellectual property, customer data, financial records, access credentials and encryption keys are among the many sensitive assets at risk."

- [Assaf Morag, Lead Threat Intelligence Analyst at Aqua Nautilus](#)

04

The Way Forward – Implement Strong Identity Protection

As organizations struggle with the expanse and complexity of Kubernetes clusters and the increased use of a variety of workloads, there is a renewed focus on adopting a robust identity-first security posture.

Identity-first security revolves around the concept of using identities and authentication for securing systems and data. Provisioning trusted identities to both users and machines (workloads, applications, and service accounts) and authenticating them helps secure access and protect distributed resources.

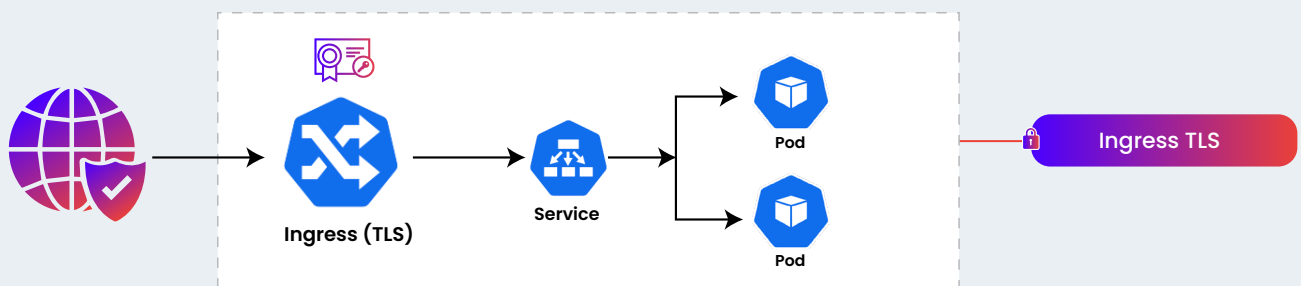
Setting identity as the primary control for access also helps align with the Zero Trust principles of explicit and continuous verification. Given these advantages, identity-first security has emerged as an effective solution for safeguarding Kubernetes environments.

SSL/TLS Certificates – The Cornerstone of Identity-First Security

Rooted in public key infrastructure (PKI), SSL/TLS is a security protocol used to identify and authenticate machines (device, application, workload) and encrypt machine-to-machine communications. Issued by trusted Certificate Authorities, SSL/TLS certificates serve as proof of identity, ensuring secure internet connections and transactions, regardless of where machines are located.

When it comes to Kubernetes, there are three critical areas where SSL/TLS can be applied for robust security.

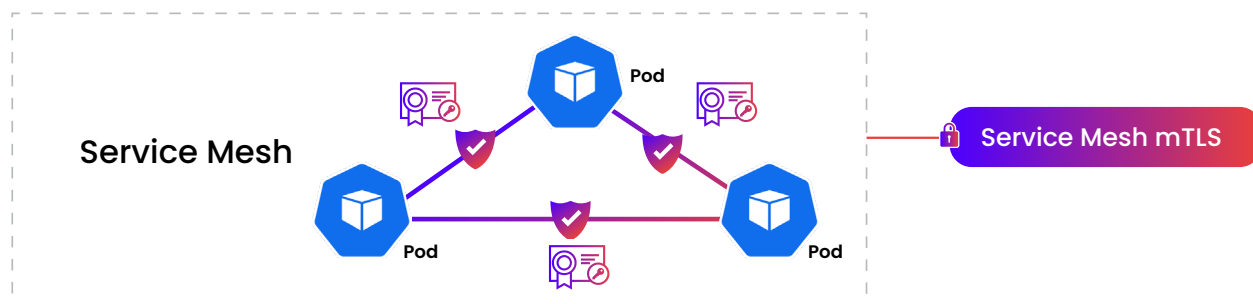
4.1. Securing the inbound web traffic to the cluster through the ingress



Controlling who can access the cluster is an effective first line of defense. As ingress points are the entry points for external traffic into a cluster, authenticating every connection at the ingress helps gain better access control, allowing only trusted parties to access cluster services. Installing SSL/TLS certificates on ingress controllers enables clients to verify the authenticity of the ingress controllers and ensure that they connect to legitimate servers and not imposters. Implementing mutual TLS (mTLS) further allows Ingress controllers to authenticate clients to ensure only trusted and authorized clients are allowed cluster access.

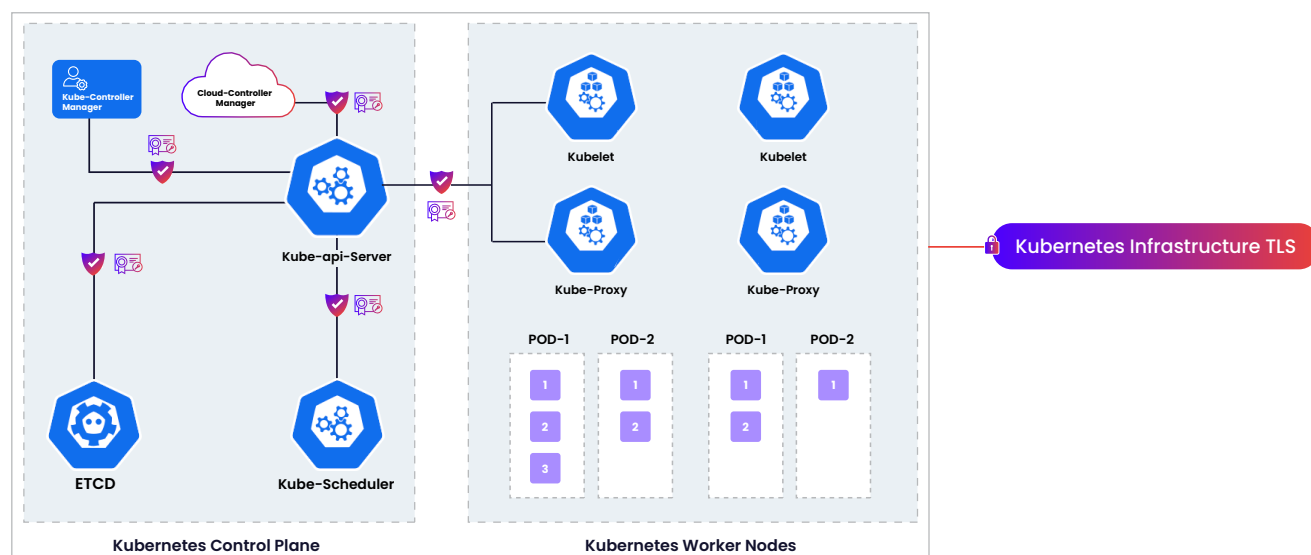
Apart from cluster access, SSL/TLS certificates also help secure north-south communication. As the web traffic entering through the ingress is in plain text, TLS helps encrypt the data to protect it from interception or tampering, establishing a secure and trusted communication channel to send and receive sensitive data, such as login credentials and other personal information.

4.2. Securing the pod-to-pod and service mesh communications inside the cluster



As mentioned earlier, microservices, containers, and pods within a cluster need to communicate with each other and with external services (often via a service mesh) to accomplish their task. Mutual TLS (mTLS) is one of the effective ways of securing the service mesh communication and preventing lateral movement. Implementing mTLS via service mesh enables microservices, pods, and containers to authenticate each other with their respective SSL/TLS certificates and allow only trusted parties to connect and communicate. Certificates can also be provisioned to the ephemeral volumes so traditional applications can read the certificate from their local volume. Besides securing access, mTLS helps encrypt the data traffic between two microservices to protect service mesh communications from being intercepted and altered.

4.3. Securing the Kubernetes infrastructure or component-to-component communications

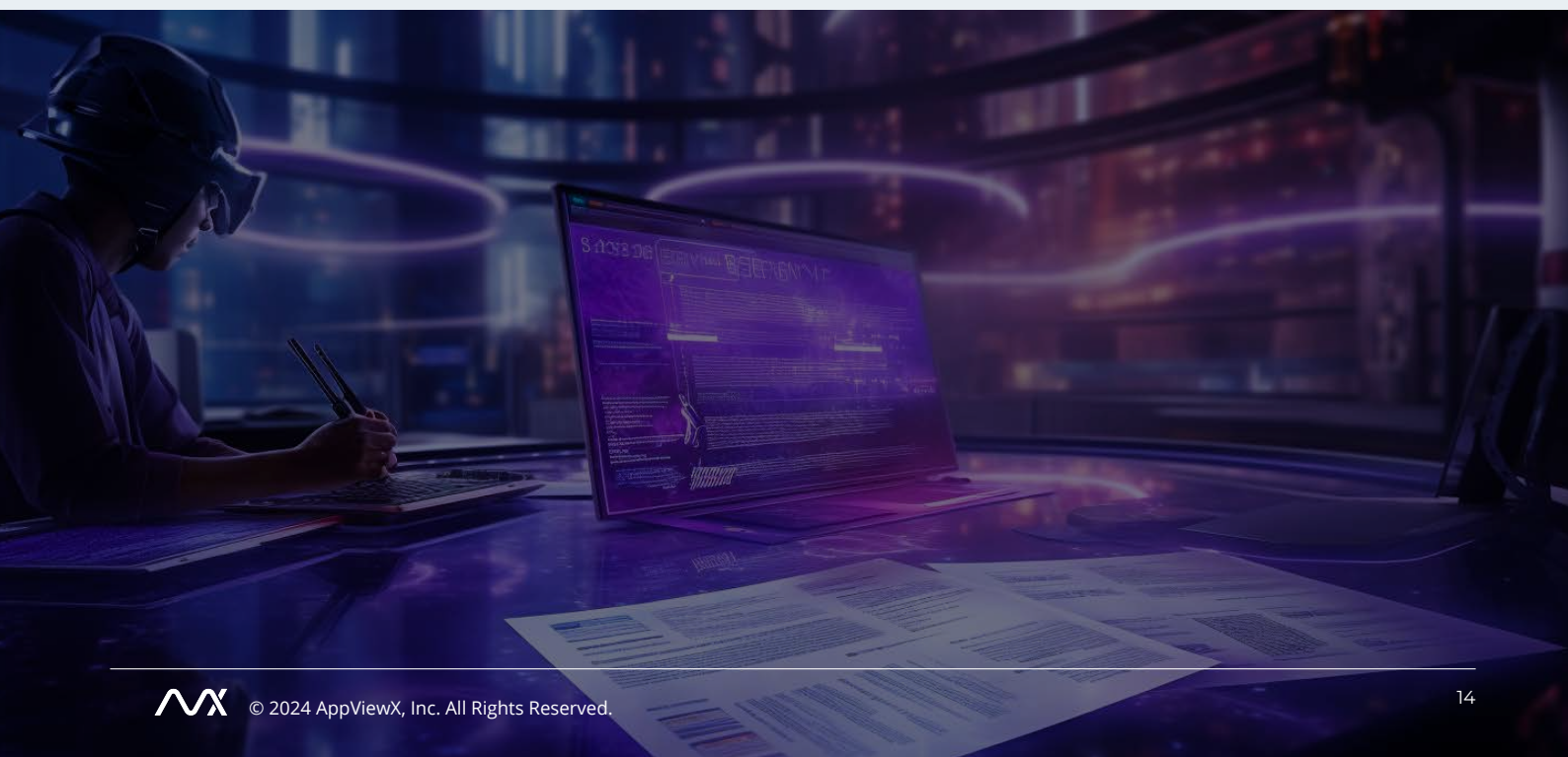


At a high level, the Kubernetes infrastructure is made of two parts: the control plane (API Server, Scheduler, Controller Manager, and etcd) and worker nodes (Kubelet, Kube-proxy, and containers).

Given how critical the control plane and the worker node components are for the smooth functioning of the cluster, it is essential to secure access to these components and safeguard their communications. This is where TLS certificates help.

TLS certificates help identify and authenticate every component before they are allowed to interact with each other. As the API Server is the primary interface for managing and interacting with the cluster, a TLS certificate is provisioned to the server, so the other components communicating with the API server can authenticate it. Further, client certificates are issued to the Controller Manager, Scheduler, etcd, Kube Proxy, and Kubelet so they can authenticate themselves to the API Server and establish a secure connection.

Along with authenticating the Kubernetes infrastructure components, TLS certificates also help safeguard secrets in the etcd database with strong encryption. This ensures that even if an attacker gains unauthorized access to the etcd data, the credentials and sensitive information will remain encrypted, providing much-needed protection from malicious actors and insider threats.



05

But... Issuing Trusted Identities Is Not Enough, Managing Them Effectively Is Key

Securing Kubernetes does not stop at provisioning SSL/TLS certificates. It is only half the solution. What is equally or rather more important is how efficiently and effectively they are managed on an ongoing basis.

Kubernetes deployments involve a large number of components, including containers, services, users, and applications, each requiring a unique digital certificate. As these certificates are often short-lived due to the ephemeral nature of Kubernetes resources, they must be continuously monitored and managed to mitigate the risk of outages, unauthorized access, and vulnerabilities caused by expired, unknown, rogue, or non-compliant certificates.

Despite the awareness and widespread usage, many PKI and Security teams are not equipped with the right tools to handle continuous monitoring and effective management of TLS certificates in DevOps pipelines and Kubernetes environments. In many cases, DevOps teams use self-signed certificates or unapproved certificate authorities to sidestep friction and then manage certificate lifecycles manually with ad hoc processes. The sheer certificate sprawl, coupled with manual processes, creates a host of additional operational complexities and risks that invariably impact cloud-native application development and delivery.

When investing in a certificate lifecycle management solution for Kubernetes, it is essential to choose solutions that enable the below best practices.

Certificate Lifecycle Management Best Practices in Kubernetes

Visibility

It is critical for DevOps, CloudOps, Platform Ops, and Security teams to have complete visibility into all certificates and their relevant data (e.g., usage, validity period, issuer, algorithms) especially because the environment is replete with short-lived containers across the cloud and on-premises.

Visibility helps to proactively identify expiring certificates, potential vulnerabilities, or issues related to certificate management. An effective way of building visibility is through continuous certificate discovery to create a comprehensive inventory with full insight into certificate-related details, including Kubernetes cluster, namespace, secrets location, chain of trust, owner, expiration dates, and crypto standards. Having this information available and up-to-date helps ensure that all the resources in the Kubernetes cluster are using valid, trusted, and compliant certificates.

Certificate Lifecycle Management Automation

In a 2023 research survey, where Kubernetes stakeholders were asked about the opportunities to improve Kubernetes infrastructure operational efficiency, automation topped the list. When it comes to SSL/TLS certificates in Kubernetes environments, developers need high-speed and on-demand certificate issuance at scale.

This requirement is best met with automation. Processes, such as certificate issuance, renewal, and provisioning, need to be automated and integrated with DevOps and CI/CD pipelines. Automation helps simplify and accelerate the whole process, preventing human error, certificate misconfigurations and outages. To further streamline certificate issuance and management, developers can be granted self-service capabilities based on RBAC (role based access control).

With effective RBAC and PKI policy enforcement, developers will not be allowed to obtain their own certificates out-of-band or using self-signed certificates, minimizing security risks.

DevOps Integration

As DevOps teams require certificates to be deployed rapidly for uninterrupted operations; the process is made easier when they can request and install certificates right from the CI/CD pipeline. This requires tight integrations between certificate management systems and DevOps/container management tools. An integrated certificate lifecycle management solution allows developers to request a certificate from any supported CA (public or private), push it to Kubernetes secrets or ephemeral pods, renew and revoke existing certificates, and delete unused certificates—all from their native DevOps CI/CD toolsets.

Policy Control

Given the abundance of certificates in Kubernetes environments, it is best to manage certificates with a well-defined, uniform PKI policy. Policy-driven management standardizes certificate processes and issuance from approved certificate authorities and minimizes the possibility of misconfigurations, vulnerabilities, and unauthorized actions.

Implementing role-based access control (RBAC) helps ensure that only authorized users have access to perform role-specific actions related to certificates, such as requesting, generating, renewing, revoking, or managing certificates. Establishing clear certificate ownership and approval workflows further helps delegate team responsibilities, improving accountability, and enabling security-controlled certificate issuance. As best practice, security teams must generate audit logs and periodic reports, documenting all critical certificate or key-related activities for audits and compliance.

Why Using Open-Source cert-manager Is Not Enough?

cert-manager is an open-source tool for certificate management. Designed for Kubernetes, cert-manager helps automate the process of issuing and renewing public and private trust SSL/TLS certificates to workloads and services.

Despite its popularity as a certificate management tool for Kubernetes, cert-manager does come with limitations. Here are key factors to consider:

- **Complexity:** cert-manager can be complex to set up and configure, especially for users new to Kubernetes. It requires a solid understanding of Kubernetes concepts and Certificate Authority (CA) infrastructure, resulting in a steep learning curve.
- **Lack of Visibility:** Certificate discovery and monitoring are integral aspects of certificate lifecycle management. They help detect issues, such as self-signed certificates and untrusted CAs. However, cert-manager is not equipped with discovery capabilities and fails to provide a comprehensive inventory or dashboard for monitoring certificates for expiration and usage across all Kubernetes clusters. Neither does it provide insights into certificate-related information, such as Kubernetes cluster, namespace, chain of trust, etc.
- **Use Case Gap:** When it comes to supporting multiple security use cases in Kubernetes, cert-manager only offers support for certificates used to secure ingress controllers and service mesh. It does not support issuance and renewal of Kubernetes infrastructure component certificates, which is highly inconvenient.
- **Limited CA Integrations:** cert-manager is not fully CA-agnostic. While it supports Let's Encrypt as a public CA and some private CAs, it offers limited support for other leading public and private CAs commonly used by enterprises. This also makes switching between CAs a significant operational challenge in the event of non-compliance or a CA compromise.

- **No Policy Control:** As discussed earlier, effective certificate management hinges on robust policies and governance. cert-manager does not incorporate features for policy enforcement. The absence of centralized auditing or governance mechanisms and the lack of support for role-based access control (RBAC) makes it difficult to standardize certificate processes across teams, prevent misconfigurations and errors, and ensure compliance.
- **No Enterprise-Grade Support:** In contrast to certificate lifecycle management solutions available in the market, cert-manager lacks dedicated enterprise-level customer support. Being an open-source project, technical assistance, bug fixes, and updates primarily rely on community contributions.










06

How AppViewX KUBE+ Simplifies Certificate Lifecycle Management in Kubernetes

AppViewX KUBE+ is a comprehensive certificate lifecycle management solution for Kubernetes environments. It provides a central solution to discover, manage, automate, and govern certificates (or machine identities) across containerized workloads and Kubernetes infrastructure. AppViewX KUBE+ brings together visibility, automation, and policy-driven control to build security into the core of DevOps pipelines and Kubernetes management.

Here is how AppViewX KUBE+ simplifies certificate lifecycle management in Kubernetes:

	Smart Discovery - AppViewX KUBE+ discovers all SSL/TLS certificates across all Kubernetes components and clusters in both on-prem and cloud environments.
	Inventory and Insights - AppViewX KUBE+ builds a comprehensive certificate inventory with full insight into certificate-related details for complete visibility.
	Certificate Lifecycle Automation - AppViewX KUBE+ automates the full certificate lifecycle from issuance to auto-renewal for certificates across Kubernetes clusters to secure ingress and pod-to-pod communications as well as kubernetes infrastructure components.
	Self-Service - AppViewX KUBE+ enables certificate self-service for developers through a fully customizable, intuitive, and user-friendly self-service portal.
	Integrations - AppViewX KUBE+ offers extensive DevOps ecosystem integrations for automating certificate lifecycles, reducing operational overhead, and enhancing Kubernetes security.
	Crypto-Agility - AppViewX KUBE+ enables full crypto-agility by allowing the configuration of policies that automate the re-issuance or renewal of certificates at scale with updated crypto-standards or issuing CA.
	PKI Policy and Compliance - AppViewX KUBE+ ensures all certificates are compliant with policies, standards, and regulations through consistent PKI policy enforcement, central audit and control, ownership hierarchy, role-based access control (RBAC), and audit logs and periodic reports.

07

Keep Your DevOps Speed Up and Kubernetes Environments Secure with Identity Protection and Effective Certificate Lifecycle Management

As containerization and Kubernetes adoption matures, the security and integrity of the environment and applications within becomes more important than ever.

Safeguarding against potential vulnerabilities, unauthorized access, lateral movement, and data breaches is imperative to uphold user trust and maintain the reliability of applications.

Given how traditional security solutions have not kept up with Kubernetes security requirements, an identity-first security framework has become a necessity for securing Kubernetes environments.

However, implementing identity-first security should not become a set-and-forget task. Organizations need to pay continuous attention to how these identities (digital certificates) are managed, especially because of massive identity sprawl. Lack of automation and a standardized approach to certificate lifecycle management will not only slow down DevOps teams but also create siloed teams and serious security risks.

Adopting a holistic certificate lifecycle management solution that offers visibility, automation, and control of the certificate ecosystem is pivotal to addressing these challenges and hardening Kubernetes security.

About AppViewX

AppViewX is trusted by the world's leading organizations to reduce risk, ensure compliance, and increase visibility through automated machine identity management and application infrastructure security and orchestration. The AppViewX platform provides complete [certificate lifecycle management](#) and [PKI-as-a-Service](#) using streamlined workflows to prevent outages, reduce security incidents and enable crypto-agility.

Need more information?

Scan QR code to schedule a free session
with one of our experts.



AppViewX Inc.,

City Hall, 222 Broadway
New York, NY 10038

info@appviewx.com
www.appviewx.com

+1 (212) 400 7541
+1 (212) 951 1146