appviewx

Solution Brief

AppViewX AVX ONE CLM for Kubernetes

Simple, Scalable, and Agile Certificate Lifecycle Management for Kubernetes Environments

Overview

Microservices and containers have completely transformed the way applications are built, deployed and run in the cloud. Advantages, such as agility, speed, portability, and scalability have made containers the de facto standard for application development and delivery today.

As more organizations adopt containerization, the popularity of Kubernetes continues to grow. Kubernetes, also known as K8s, is an open-source platform that helps organizations effectively manage containerized applications at scale.

Kubernetes has been credited for saving DevOps teams significant time and effort, ensuring application availability and stability, and allowing DevOps to remain focused on agile development at scale. Given the benefits, a majority of organizations today run several clusters across multiple cloud environments.

As with any new cloud-native technology, the adoption of containers and Kubernetes has given rise to a whole new attack surface and several unforeseen security challenges. As traditional security controls are not built for the cloud, it's a constant struggle for InfoSec teams to gain visibility into Kubernetes clusters and ensure security across multiple clouds.

What Makes Securing Kubernetes Difficult?

Kubernetes Environments Are Complex

Kubernetes environments are highly distributed and dynamic in nature. There are a large number of small moving components that need to be continuously monitored and protected. In addition, containers in Kubernetes are ephemeral with lifespans as short as a couple of minutes. They are also spun up / down, created / deleted / rescheduled based on demand. As a result, clusters constantly grow and shrink, making it difficult to secure. What's also important to note is that Kubernetes isn't inherently secure. While the platform provides built-in security features, they are very limited. Hardening a cluster will need additional security solutions that can protect all layers of the Kubernetes stack.

Disconnected Processes and Team Silos

Enterprises today run hundreds or thousands of Kubernetes clusters that are typically managed by multiple teams using disparate processes and tools, resulting in inconsistencies and non-compliance. Another challenge is the misalignment between DevOps, CloudOps, and InfoSec teams. While DevOps and CloudOps need to move quickly, InfoSec needs to ensure security and compliance. As security tools tend to cause delays and slow down release cycles, developers often see security as an inhibitor and resort to workarounds that put enterprise security at risk. On the other hand, InfoSec teams have little visibility or control over DevOps security practices, leading to blind spots, vulnerabilities, and compliance violations.

To make the most out of Kubernetes deployments, it is essential to follow a cloud-native security strategy, which involves securing the cloud infrastructure and applications from the beginning of the development lifecycle. It is recommended to implement security solutions that can navigate the complexity of Kubernetes and protect every component, without impacting speed or agility.

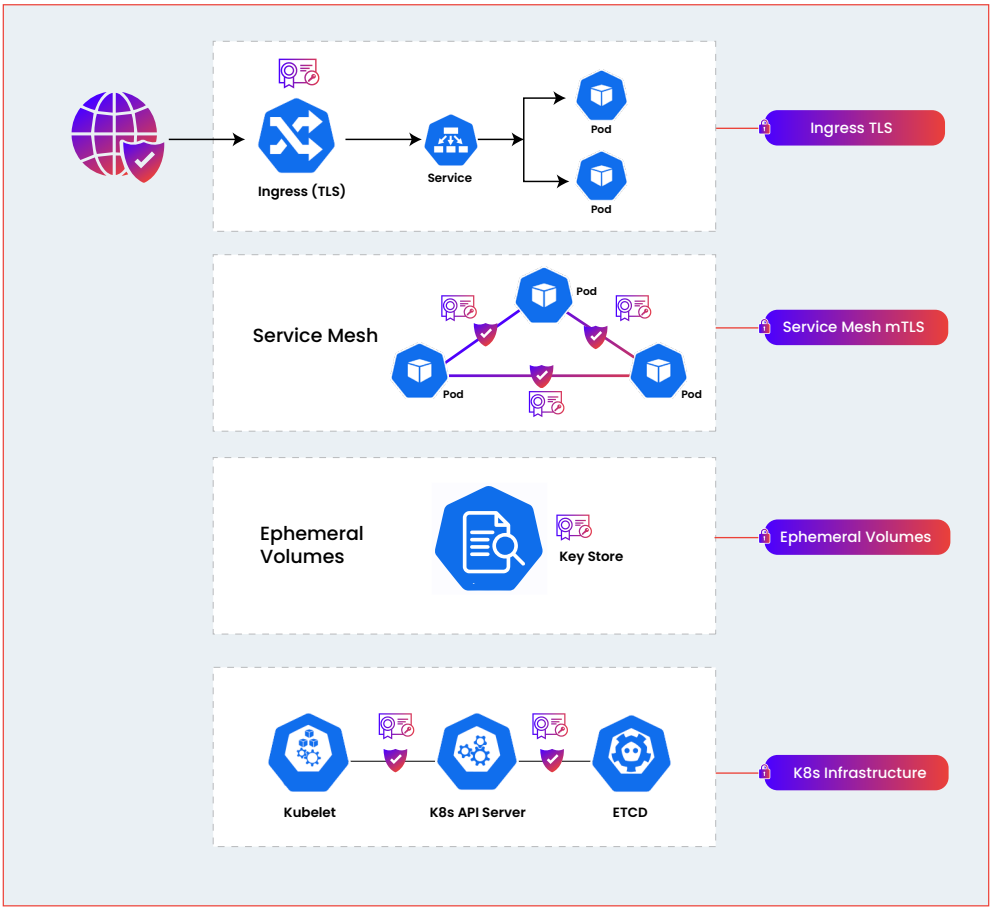
Why TLS Is a Must for Securing Kubernetes

TLS certificates are an integral part of application security, especially in the cloud. They help protect all workloads, applications, and machines with unique identities, reliable authentication, and strong encryption.

When it comes to setting up a Kubernetes cluster, TLS is fundamental to identifying all the components within the cluster, enabling authentication between them, and securing their communications.

In fact, Kubernetes itself recommends TLS security for its components - *"TLS should be enabled for every component that supports it to prevent traffic sniffing, verify the identity of the server, and (for mutual TLS) verify the identity of the client."*

TLS Certificate Use Cases in Kubernetes Environments:



Three critical areas where TLS helps secure in Kubernetes environments, include:

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

As ingress points serve as entry points for external traffic into a cluster, securing them is crucial to ensure that only authorized traffic reaches the services, protecting them from attacks and breaches. Implementing TLS is one of the effective ways of securing the ingress and the cluster from potential threats.

When TLS server certificates are installed on ingress controllers, they help clients verify the authenticity of the ingress controllers and ensure that they are connecting to legitimate servers and not imposters, safeguarding their network connection from man-in-the-middle attacks.

If multiple users and teams use the ingress controller for various purposes, a more effective approach would be to implement mutual TLS (mTLS) at the ingress. This allows both the client and the ingress controller to authenticate each other with their respective TLS certificates, providing an extra layer of security and eliminating unauthorized access.

Securing north-south communication is another significant area where TLS helps. As the web traffic entering through the ingress is in plain-text, TLS helps encrypt the data to protect it from interception or tampering. TLS establishes a secure and trusted communication channel so sensitive data, such as login credentials, personal information, and other confidential data, aren't exposed.

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

In a Kubernetes cluster, different microservices, containers, and pods communicate with each other and with external and internal endpoints often via service mesh to accomplish their defined task. In such an interconnected set up, even if a single microservice is breached, the impact is multifold as the breach can quickly propagate to all the other pods and containers, connected to the compromised microservice.

mTLS is one of the effective ways of securing the service mesh communication as it enables microservices, pods, and containers to authenticate each other with their unique identities, and allow only trusted parties to connect and communicate.

For example, when two pods in a service mesh want to communicate with each other, mTLS enables both the client pod (requestor) and the server pod (responder) to authenticate each other by verifying their respective TLS certificates to establish a secure connection. Certificates can also be provisioned to the ephemeral volumes so traditional applications can read the certificate from their local volume.

Apart from securing access, mTLS also helps encrypt the data traffic between two microservices. This helps protect the service mesh communication from being intercepted and altered.

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 and worker nodes.

The control plane in Kubernetes is the brain or command center that manages and controls the operations of a cluster. It includes an API Server, Scheduler, Controller Manager, and etcd that work together to ensure that the workloads are running as expected. A worker node, on the other hand, consists of two components - Kubelet and Kube-proxy - that communicate with the control plane to run pods and the applications within them smoothly.

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.

Every component in the control plane and a worker node has different levels of access to the cluster. TLS certificates help identify and authenticate every component before they are allowed to interact with each other.

As the API Server is the main entry point to the entire 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 encrypt the secrets at rest. Etcd is a distributed key-value store that stores information about the configuration and state of all clusters, pods, and the services within them.

In general, the information (secrets) in the etcd database is accessible via the Kubernetes API, which attackers may exploit to gain visibility into the state of your cluster. TLS certificates help prevent this by encrypting secrets at rest. This ensures that even if an attacker gains unauthorized access to the etcd data, they won't be able to easily intercept the sensitive information, providing the much-needed layer of protection for the cluster.

Certificate Lifecycle Management Challenges in Kubernetes Environments

Lack of Visibility:

Lack of Visibility Into Kubernetes Certificates

It's critical for InfoSec teams to have complete visibility of all the certificates to ensure that all the components in the Kubernetes cluster are using valid, trusted, and compliant certificates. However, with DevOps teams often using self-signed certificates or getting certificates from their preferred CAs independently to avoid delays, InfoSec teams quickly lose oversight and control. This creates security blind spots in the form of unmanaged, rogue, and non-compliant certificates.

Monitoring Certificates for Expiry

As Kubernetes environments require a large volume of short-lived certificates, renewals, rotations and re-provisioning are highly frequent. However, it takes tremendous effort to track certificates and keep a tab on their varying validities through Outlook calendars and spreadsheets. Missing certificate expiry becomes routine, leading to application outages, security weaknesses, and service disruptions.

Inefficient Processes and Lack of Integrations:

Manual Processes Are Neither Fast nor Scalable

Developers need high speed certificate issuance at scale. However, the traditional process for requesting and issuing a certificate involves raising a ticket, seeking multiple approvals, and waiting for days. Even after a certificate is issued, developers will still need to install and bind certificates manually, which eats into developers' productivity and increases the possibility of certificate misconfigurations and outages. Because manual processes are painfully slow, developers often dodge certificate management best practices in favor of speed, amplifying cyber risks.

No Integrations Support

As DevOps teams require certificates to be deployed rapidly for uninterrupted operations, it would be a lot easier if they could request and install certificates right from the CI/CD pipeline. This requires tight integrations between certificate management systems and DevOps/ container management tools. Ad hoc CA tools and internal PKI set ups (i.e. Microsoft CA) are not equipped to support any of these integrations, which is not conducive to DevOps.

Non-compliance and Lack of Control:

Lack of Standard Compliance and Policy Control

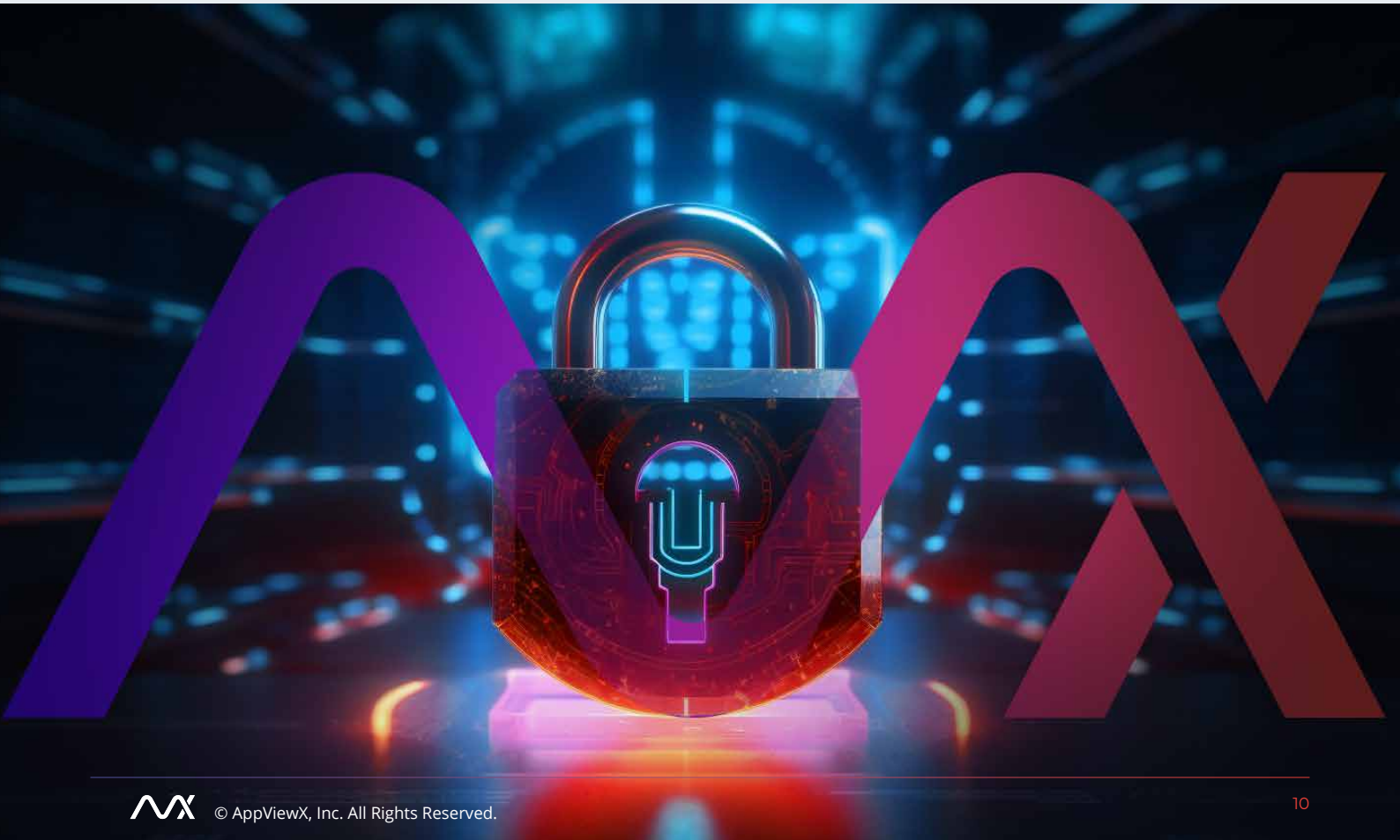
Manual or Ad hoc processes provide no means to enforce a uniform PKI policy. Lack of well-defined policies and certificate sprawl often lead to weak crypto standards, use of vulnerable self-signed certificates and certificates procured from unapproved CAs, and misconfigurations—all of which introduce security risks and non-compliance issues. Compounding the problem is the lack of clear certificate ownership and approval workflows that lead to expired certificates, unauthorized access, and rogue certificate issuance.

No Central Audit and Governance Across Kubernetes

As there are multiple teams managing certificates in a single cluster, all certificate and key related activities must be closely governed to avoid security lapses and simplify auditing. When certificates are managed manually, it is difficult to log and monitor user and certificate related actions. The lack of governance impacts both auditing and threat remediation.

Lack of Role-Based Access Control (RBAC)

Providing conditional access to certificates is essential to ensure that only authorized users have access to perform role-specific actions related to certificates, such as requesting, generating, renewing, revoking, or managing certificate authorities (CAs). However, manual processes and adhoc tools do not support enforcing RBAC, increasing the risk of mismanaged certificates and human errors.



Simplify and Modernize Certificate Lifecycle Management for Kubernetes with AVX ONE CLM for Kubernetes

AVX ONE CLM for Kubernetes 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. By bringing together visibility, automation, and policy-driven control, AVX ONE CLM for Kubernetes bakes security into the core of DevOps pipelines and Kubernetes management.

1. Visibility: Complete Visibility of All Certificates Across Kubernetes Environments

Smart Discovery

AVX ONE CLM for Kubernetes automatically discovers all SSL/TLS certificates installed across all Kubernetes clusters and containerized workloads in both on-prem and cloud environments. It also detects and documents granular Kubernetes data, such as the cluster, namespace, secrets, etc.

Inventory and Insights

Once the discovery is complete, AVX ONE CLM for Kubernetes automatically builds a comprehensive certificate inventory with full insight into certificate-related details including Kubernetes cluster, namespace, secrets location, chain of trust, owner, expiration dates, and crypto standards. All certificates and their information are presented in a central dashboard view for single-pane-of-glass visibility. The inventory can be segmented into groups and mapped to enterprise-wide Kubernetes teams, such as by namespace to simplify certificate management across any number of clusters.

Reporting and Notifications

AVX ONE CLM for Kubernetes allows you to build custom certificate reports based on any critical attribute or certificate metadata, such as expiry dates, issuing CA, etc. These reports help PKI, DevOps, CloudOps, and Platform Ops teams stay up to date on all certificates across their environments, and monitor for upcoming expirations and levels of compliance. AVX ONE CLM for Kubernetes also comes with built-in alerts for expiry that are delivered via emails or SNMP traps to help teams stay on top of impending renewals. In addition, AVX ONE CLM for Kubernetes also detects and reports on self-signed certificates used in the cluster.

2. Automation: End-to-End Certificate Lifecycle Automation, Self-Service, and Integrations

CLM Automation

AVX ONE CLM for Kubernetes automates the full certificate lifecycle from issuance to auto-renewal for all certificates across Kubernetes clusters to secure ingress and pod-to-pod communications as well as kubernetes infrastructure components. The automation workflows are fully customizable and can trigger any number of approvals, notifications and change controls. AVX ONE CLM for Kubernetes also can automatically provision certificates to the ephemeral volumes, which can then be used by pods for secure communications.

Self-Service

AVX ONE CLM for Kubernetes provides a fully customizable, intuitive, and user-friendly self-service portal with role based access control (RBAC) to enable teams to easily request and manage certificates on their own without any dependencies. All certificate requests can be validated/ authorized by SecOps using approval workflows and with policy enforcement.

Integrations

AVX ONE CLM for Kubernetes offers extensive DevOps ecosystem integrations for automating certificate lifecycles, reducing operational overhead, and enhancing Kubernetes security. These integrations allow DevOps teams to request certificates from any supported CA (public or private), push it to Kubernetes secrets or ephemeral pods, which can then be associated with the applications, renew and revoke existing certificates, and delete unused certificates—all from their native DevOps CI/CD toolsets. The integration ecosystem includes:

- Public CAs such as Digicert, Entrust, GlobalSign, and Sectigo
- Private CAs such as AVX ONE PKIaaS, Microsoft CA, EJBCA, and Amazon Certificate Manager (ACM)
- Secrets Managers such as HashiCorp Vault
- Integrations with HSM/ IAM including Entrust, Fortanix, Utimaco, Safenet, Thales, Ldap, and Radius
- TLS keystores
- ITSM such as ServiceNow CMDB and BMC Remedy
- Kubernetes and Managed K8s deployments such as hybrid/on-prem (Openshift, Tanzu, Rancher) and cloud provider solutions (EKS, AKS, GKE)
- ServiceMesh such as Istio and Linkerd
- DevOps and CI/CD tools such as Terraform, Ansible, SCM, and Jenkins
- ChatOps: Email approvals, Slack, and Pagerduty

Crypto-Agility

The ability to quickly adapt and respond to cryptographic threats or failures is crucial to staying secure and compliant. AVX ONE CLM for Kubernetes allows you to easily configure policies to automate the re-issuance or renewal of certificates at scale with updated crypto-standards, enabling full crypto-agility. You can also seamlessly change Certificate Authorities using the simple “CA Switch” feature without causing any operational disruption in the event of non-compliance etc.

3. Control: PKI Policy & Governance

PKI Policy

- AVX ONE CLM for Kubernetes offers a central console to audit and control all certificates across hundreds of Kubernetes clusters in hybrid multi-cloud infrastructures
- AVX ONE CLM for Kubernetes allows you to define and enforce standardized PKI policies for certificates across all Kubernetes clusters. You can simplify policy and maintain control over certificate validity, approved public and private CAs, key size, algorithm, and more to eliminate rogue and non-compliant certificates.
- AVX ONE CLM for Kubernetes allows establishing ownership hierarchy with an approval chain and well-defined roles to ensure easy and compliant certificate management across various Kuberentes teams within an organization
- To enable seamless self-service for DevOps teams while reducing the risk of human errors and unauthorized certificate actions, you can enforce granular role-based access control (RBAC) and approval workflows. RBAC helps provide role-specific permissions, improve accountability, and enable security-controlled certificate issuance.

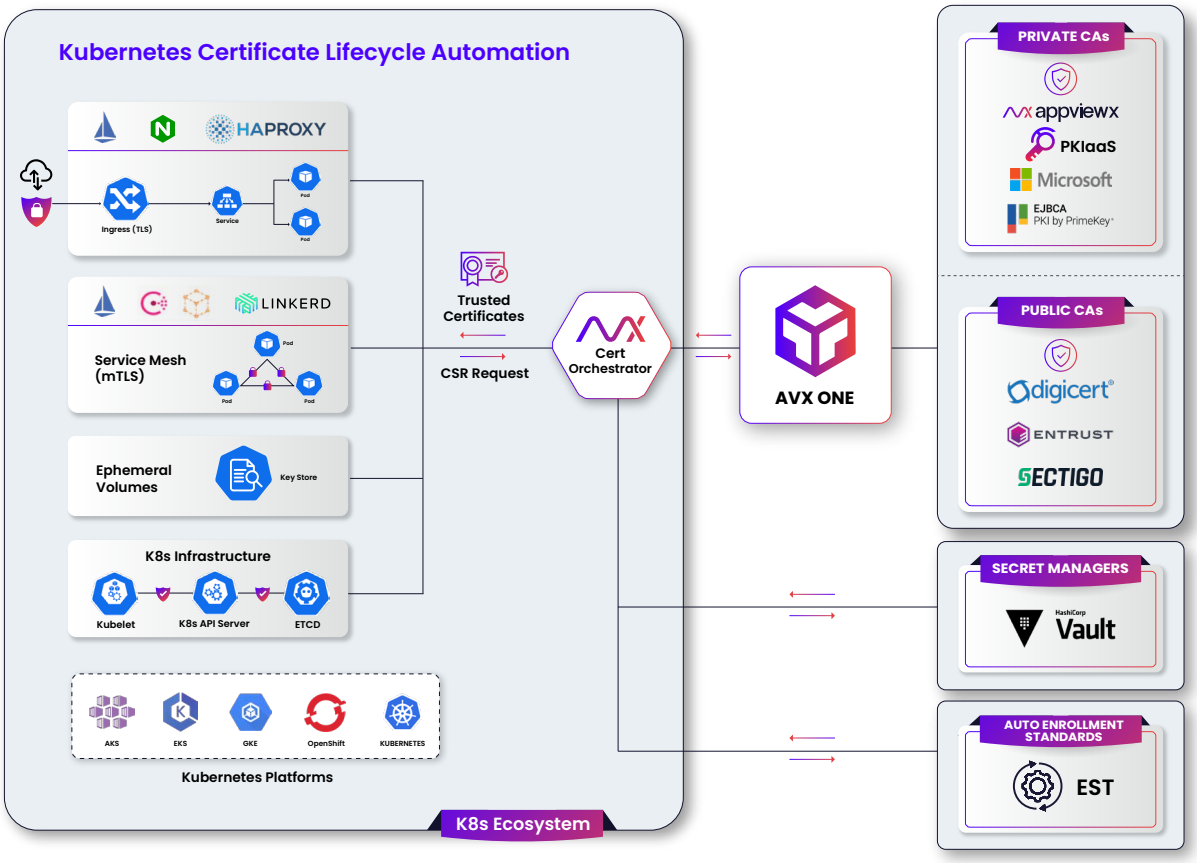
Governance

AVX ONE CLM for Kubernetes can generate audit logs and periodic reports of certificate usage for easier audits and regulatory compliance. You can group certificates and keys based on specific business use cases, create audit logs to document all critical certificate or key-related activities, and generate reports to simplify audits and ensure compliance.

Business Benefits of AVX ONE CLM for Kubernetes

- **Complete Control Over the Certificate Ecosystem** - Easily manage the certificates spread across containerized workloads and hosted clusters or infrastructures regardless of the cluster size or complexity.
- **No Security Blind Spots** - Maintain complete visibility into the cloud-native certificate inventory to monitor and proactively remediate misconfigurations or non compliant certificates.
- **No Application Outages** - Ensure all certificates are renewed on time and applications are always up and running, with timely alerts, self-service, automation workflows, and auto-renewals.
- **Continuous Compliance** - Ensure all certificates are compliant with regulations, such as GDPR, PCI DSS, and HIPAA, with consistent policy enforcement, audit logs, and periodic reports.
- **High Scalability** - Instantly scale up or down to meet cloud-native certificate requirements without investing in PKI hardware or software resources.
- **DevOps-InfoSec Alignment** - Bring security up to speed with DevOps. Break the silos between DevOps and InfoSec teams with end-to-end CLM automation and seamless integrations.
- **Speed, Agility, Resiliency** - Accelerate DevOps efficiency while ensuring security of Kubernetes environments, with visibility, certificate lifecycle automation, and consistent policy enforcement.

AVX ONE CLM for Kubernetes Architecture



AVX ONE CLM for Kubernetes Consumption Models

SaaS

Available as a service, AVX ONE CLM for Kubernetes is fully managed and updated by AppViewX. Customers can directly set up an account and start using it. For connecting to the non-public corporate network segments without poking a hole into the corporate firewall, AppViewX provides a Cloud Connector. The Cloud Connector connects to the Kubernetes clusters through AVX ONE CLM for Kubernetes component, called Cert Orchestrator, that is installed within the cluster for certificate management.

On-Prem

The CLM automation capabilities of AVX ONE CLM for Kubernetes can also be deployed within a customer's environment in hypervisor-based VMs, private clouds, or public clouds using AWS, GCP, Microsoft Azure, and others. AVX ONE CLM for Kubernetes can be installed on any virtual machine instance running CentOS or RHEL operating system.

Managed Kubernetes

As the AVX ONE platform is a Kubernetes-based application, it can also be installed in a managed Kubernetes environment like EKS, AKS, GKE, RedHat Openshift, Rancher, and others.

AppViewX Inc.,

City Hall, 222 Broadway
New York, NY 10038

info@appviewx.com
www.appviewx.com

+1 (206) 207-7541
+44 (0) 203-514-2226