

ARMO

The complete guide to_

Achieving compliance with Kubernetes

Learn about Kubernetes compliance challenges, consequences of non-compliance, and get guidance on maintaining a secure and compliant cloud environment in a dynamic Kubernetes setup.

 See ARMO in Action



Ben Hirschberg
CTO and Co-Founder, ARMO

Table of contents

Introduction	5
Regulatory frameworks and standards	6
Kubernetes attack surface	7
Challenges of achieving compliance in Kubernetes	7
Why is compliance in Kubernetes challenging?	8
Complex architecture	8
Ephemeral state	8
Lack of holistic visibility	9
Scalability	9
What are the limitations of existing Kubernetes compliance solutions?	9
Lack of Kubernetes-Specific features	9
Limited automation	10
Limited integration	10
Overcoming limitations of current solutions	10
Tools	11
Best practices	11
Consequences of non-compliance	12
Summary	13
Compliance Score: how ARMO simplifies compliance assessment	14
Control Compliance Score: a detailed assessment	14
Framework Compliance Score: a holistic view	15
Key takeaways	15
Achieving compliance with cloud-managed Kubernetes	16
Comparing the compliance status of different Kubernetes distributions	16
Google Kubernetes Engine (GKE)	16
Amazon Elastic Kubernetes Service (EKS)	17
Azure Kubernetes Service (AKS)	17
Achieving compliance with different Kubernetes distributions: best practices	19
Regular security assessments, vulnerability scans, and misconfigurations	19
Role-Based Access Control (RBAC) and audit logging	19
Adhering to industry-specific regulations	19
Encrypting data in transit and at rest	20
Establishing a robust incident response plan	20
Continuous scanning, monitoring, and remediation for Kubernetes compliance	20
ARMO Platform	21

Kubernetes compliance under SOC 2 **23**

- Understanding SOC 2 compliance for Kubernetes clusters** **23**
- Kubernetes security features and tools for meeting SOC 2 requirements** **24**
 - Network segmentation of publicly accessible and internal components 25
 - Firewalling of ingress and egress traffic between applications 25
 - Encryption of data in transit 26
 - Enforce workload isolation to limit blast radius 26
- Guidelines for ensuring SOC 2 compliance over time** **27**
 - Continuous logging, monitoring, and automated audit scanning 27
 - Preventing misconfigurations 27
 - Traffic logging 28
 - Automated audit scanning 28
- Key Takeaways** **29**

Kubernetes compliance under GDPR **30**

- GDPR principles and requirements on data privacy** **30**
- How GDPR applies to Kubernetes compliance** **31**
 - Access control and authentication 31
 - Data management at rest and in transit 31
 - Network segmentation 32
 - Auditing and continuous monitoring 32
- Best practices for Kubernetes compliance under GDPR** **32**
 - Set up Role-Based Access Control 32
 - Use encryption to secure data at rest and in transit 33
 - Implement network segmentation and firewalling 33
 - Setting up continuous logging, monitoring, and alerting 34
- Key Takeaways** **34**

Kubernetes compliance under HIPAA **35**

- Security guidance frameworks — HIPAA** **35**
- The HIPAA privacy rule** **36**
 - Seven principles for HIPAA compliance 36
 - HIPAA compliance strategies 36
- Privacy protection frontier with internals in mind** **37**
- RBAC and policy enforcement** **38**
- Policy engines: Advantages and drawbacks** **38**
- OPA-based techniques for policy enforcement** **39**
- Pod security standards** **39**
- Using OPA with PSS** **40**
- Kyverno’s policy-as-code approach** **40**
- ARMO Platform** **41**

Key Takeaways	42
----------------------	-----------

Kubernetes compliance under ISO 27001 **43**

What is ISO 27001?	43
---------------------------	-----------

Kubernetes: sensitive workloads and attack surface	43
---	-----------

ISO 27001 and Kubernetes	44
---------------------------------	-----------

Section 4: context of the organization	44
--	----

Section 5: leadership	44
-----------------------	----

Section 6: planning	45
---------------------	----

Section 7: support	45
--------------------	----

Section 8: operation	45
----------------------	----

Section 9: performance evaluation	45
-----------------------------------	----

Section 10: improvement	45
-------------------------	----

Kubernetes security features and tools that support ISO 27001 compliance	46
---	-----------

Best practices for achieving Kubernetes compliance under ISO 27001	47
---	-----------

Define a security policy	47
--------------------------	----

Implement access controls	48
---------------------------	----

Use network security measures	48
-------------------------------	----

Employ encryption techniques	48
------------------------------	----

Stay on top of the latest security best practices	48
---	----

Key Takeaways	49
----------------------	-----------

Kubernetes compliance under PCI **50**

PCI DSS requirements	50
-----------------------------	-----------

Kubernetes security features and tools for meeting PCI DSS requirements	51
--	-----------

Implementing network segmentation	52
-----------------------------------	----

Firewalling of inbound and outbound traffic between applications	52
--	----

Encrypting data in transit	53
----------------------------	----

Restricting cardholder data to a Need-to-Know basis	53
---	----

Monitoring and auditing best practices for PCI compliance	54
--	-----------

Continuous logging, monitoring, and automated audit scanning	54
--	----

Preventing misconfigurations	54
------------------------------	----

Traffic logging	55
-----------------	----

Automated audit scanning	55
--------------------------	----

Key Takeaways	55
----------------------	-----------

About ARMO **56**

ARMO Platform	56
----------------------	-----------

Introduction

Kubernetes is a leading open-source platform for automating containerized applications' deployment, scaling, and management. With the growing adoption of cloud, hybrid, and multicloud environments, the topic of **Kubernetes compliance** has become increasingly pertinent. Kubernetes compliance means ensuring that the platform and its components adhere to applicable regulations and standards. With a rapidly growing attack surface in modern cloud environments, the emphasis on compliance has increased among Kubernetes users.

This whitepaper will explain what compliance is in Kubernetes, the challenges of achieving it, the consequences of non-compliance and will cover various compliance types (SOC 2, GDPR, HIPPA, ISO27001). It will then explore how to achieve compliance in this dynamic, ephemeral environment. You will gain insights, guidance, and tool recommendations to help maintain a secure and compliant Kubernetes infrastructure.

Understanding compliance in Kubernetes

Compliance in Kubernetes applies to various aspects of the platform, including **security**, data privacy, network security, and incident response. The goal of Kubernetes' compliance requirements is to minimize the risk of security breaches and ensure that sensitive data is protected. This section will present the leading regulatory **frameworks and standards** that apply to Kubernetes in specific contexts, and discuss the significance of the Kubernetes attack surface.

Regulatory frameworks and standards

Several regulatory frameworks and standards apply to Kubernetes, depending on the industry and location of the organization. Some examples include:

SOC 2: The Service Organization Control (SOC) 2 framework applies to organizations that provide cloud-based services. It addresses data security, availability, processing integrity, and confidentiality standards.

PCI DSS: The Payment Card Industry Data Security Standard (PCI DSS) applies to organizations that handle credit card information, and sets standards for securing sensitive data. PCI DSS requires regular security assessments.

HIPAA: The Health Insurance Portability and Accountability Act (HIPAA) applies to organizations in the healthcare industry. It is concerned with the protection of patient data, including requirements for access controls, data encryption, and incident response, and establishes standards pertaining to these factors.

NIST SP 800-53: The National Institute of Standards and Technology (NIST) Special Publication (SP) 800-53 regulates the security of federal information systems and organizations. Its scope includes requirements for incident response, access controls, and data encryption.

GDPR: The General Data Protection Regulation (GDPR) is a regulation in EU law for data protection and privacy, applying to all individuals within the European Union. Protecting personal data—including requirements for data breach notification, right to erasure, and data portability—are within its domain.

ISO 27001: The International Organization for Standardization (ISO) 27001 is an international standard for information security management that addresses information security risks, including requirements for incident management, access controls, and data encryption.

Kubernetes attack surface

The attack surface of Kubernetes refers to the various areas of a [Kubernetes cluster](#) that are vulnerable to an attack. These areas, or attack surfaces, can be grouped as follows:

Kubernetes control plane

Nodes

Network

Add-ons, such as [ingress controllers](#), [service meshes](#), and monitoring solutions

As Kubernetes adoption has grown, so have new use cases which drive more complex architectures. This level of adoption has also increased the attack surface of Kubernetes clusters and led to it becoming more diverse and harder to control. This is the direct result of increasing numbers of Kubernetes clusters being deployed in multicloud and hybrid-cloud environments. Thus, organizations must implement robust security controls and continuous monitoring in order to protect their Kubernetes stack from potential attacks.

Compliance frameworks play a crucial role in securing Kubernetes clusters. These frameworks provide guidelines and best practices that help organizations establish robust security controls and ensure adherence to industry regulations. By following compliance frameworks, organizations can implement essential security measures, such as access controls, network policies, and data encryption, to protect sensitive information and prevent unauthorized access. However, achieving compliance in Kubernetes can be quite challenging due to its dynamic and distributed nature. The inherent complexity of Kubernetes, with its multitude of components and the need to manage configurations and updates across clusters, adds an additional layer of difficulty to ensuring compliance. Nevertheless, organizations must rise to

Challenges of achieving compliance in Kubernetes

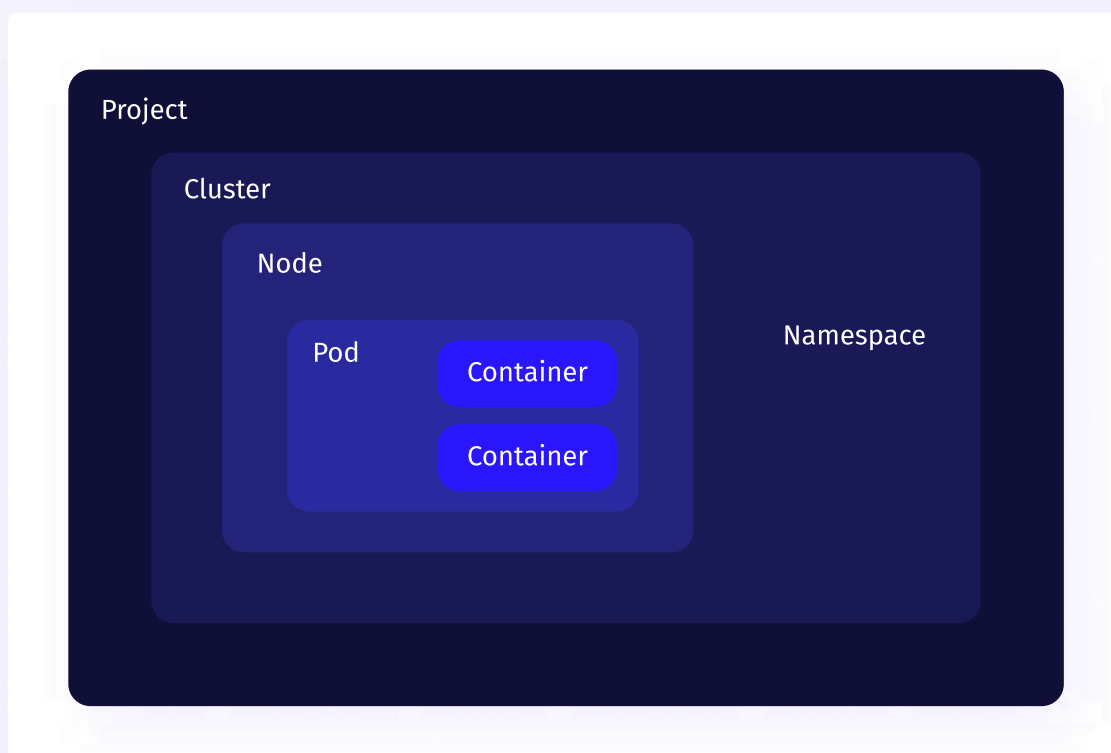
Compliance in Kubernetes is a complex beast, due to the architectural characteristics of the cloud and the limitations of current compliance tools. This section will first discuss difficulties related to the dynamic and ephemeral features of Kubernetes and then look at existing compliance tools' limitations, before suggesting ways to overcome these limitations.

Why is compliance in Kubernetes challenging?

Achieving compliance in a dynamic, ephemeral environment such as Kubernetes can be challenging for a number of reasons, which deserve a closer look.

_Complex architecture

Kubernetes clusters are commonly used for complex applications that use [microservices](#). Their architectural complexity makes compliance across the entire environment a challenge; there are often multiple interconnected components to consider.



_Ephemeral state

Pods and containers in a Kubernetes cluster are ephemeral, meaning that they are typically created and destroyed quickly and frequently. This transient state makes it challenging to maintain consistent compliance across the entire stack, as resources may be added or removed without proper oversight.

_Lack of holistic visibility

Kubernetes creates a complex stack with applications running both in the cluster and in the underlying cloud infrastructure. There are widely adopted tools available to improve visibility, such as [Prometheus](#) for monitoring, [Grafana](#) for visualization, and initiatives such as [OpenTelemetry](#) to create a unified platform. However, these tools are limited in terms of their focus, and creating holistic visibility often remains elusive. This creates challenges in identifying and addressing compliance issues in a timely manner.

_Scalability

One of the critical features of Kubernetes is its ability to scale resources automatically based on demand. Although it is a compelling feature, finding security issues in a thousand-node production environment is challenging.

What are the limitations of existing Kubernetes compliance solutions?

While Kubernetes poses several compliance challenges, the platform actively fosters collaboration with a vibrant, open-source community, and stays attuned to industry trends. It's reasonable to assume that in the coming years, compliance management for Kubernetes will be improved and simplified. Current compliance solutions in the Kubernetes ecosystem have several limitations, including a lack of Kubernetes-specific features, limited automation, and restricted integration.

_Lack of Kubernetes-Specific features

Many compliance solutions are designed for traditional cloud infrastructure environments. As a result, their capacity to address Kubernetes-native features—such as extensions with [custom resources](#) or [RBAC controls](#)—is limited.

_Limited automation

Compliance solutions that rely on manual inspections and audits to ensure compliance can prove to be time-consuming and error-prone. In an ephemeral environment such as Kubernetes they can also prove to be futile. In addition, without automation and continuous controls, ensuring compliance in Kubernetes is onerous.

_Limited integration

Some compliance solutions may be siloed and integrate poorly with other tools and systems used in the Kubernetes ecosystem. This lack of integration between monitoring, logging, auditing, and node-level container runtimes tools may result in suboptimal visibility and can slow down identification of security breaches and patching of them.





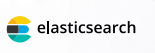
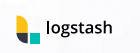




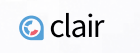




Overcoming limitations of current solutions

In order to overcome these limitations, organizations should adopt compliance solutions designed explicitly for Kubernetes environments. These solutions should include features such as automated compliance checks, Kubernetes-specific auditing, and integration with other tools and systems used in the Kubernetes ecosystem. Please note that relying on compliance tools designed for a cloud environment does not guarantee a sufficient feature set to ensure Kubernetes cluster compliance.

Here are three important things to consider when looking for ways to ensure compliance in a dynamic, complex Kubernetes environment:

_Tools

As opposed to a single solution, it is recommended to deploy a number of tools that should be integrated into your overall compliance framework:

Category	Solution Providers
Code compliance	 Open Policy Agent  Kubescape + Kube-compliance
Admission controllers	 Kyverno  GATEKEEPER + Built-in Kubernetes feature
Auditing & logging	 elasticsearch  logstash  kibana  fluentd  Prometheus
Image scanning	 aqua trivy  clair  Kubescape
Networking	 Calico  cilium  Istio

_Best practices

In addition to selecting the most effective tools, ensuring best practices is also an important part of maintaining Kubernetes security:

- Verify which regulations apply to your organizations prior to deployment
- Define and distribute documentation of policies and standards for your organization
- Automate compliance checks and remediation efforts
- Regular team training sessions and updates
- Use Kubernetes-specific admission controllers
- Use Kubernetes-specific auditing and logging tools
- Enable, automate and monitor container image scanning
- Leverage historical monitoring data to perform monthly audits and quarterly reviews

Consequences of non-compliance

Non-compliance in a Kubernetes environment can create avoidable risks and lead to serious consequences, including:

Security vulnerabilities: Non-compliance with security standards and regulations can leave a Kubernetes environment open to attacks and breaches. This can cause sensitive data to be compromised, resulting in financial loss and damage to an organization's reputation.

Compliance penalties: Organizations that are found to be non-compliant with regulatory standards may be subject to fines, liabilities, and legal action.

Difficulty in passing audits: Organizations will not pass audits and maintain certifications if they are not compliant. This can result in lost business opportunities. Non-compliant organizations will not receive the compliance certifications (e.g. PCI-DSS, SOC2, HIPAA, and NIST) necessary for doing business.

Difficulty in detecting and responding to incidents: Non-compliant clusters can cause failures when it comes to detecting and responding to security incidents, as there may be insufficiencies in monitoring and logging.

Struggle to maintain a competitive edge: Organizations may lose their competitive advantage in the market if they are not compliant, as they may not be able to meet the baseline compliance requirements of potential customers and partners.

To mitigate these risks and consequences, organizations should adopt a comprehensive compliance strategy that includes regular audits, automated compliance checks, and ongoing monitoring of the Kubernetes environment.

Intro's Summary

Compliance in a Kubernetes environment is a complex and ongoing process. The dynamic and ephemeral nature of clusters, combined with the growing attack surface in modern cloud environments, makes it challenging to maintain a consistent state of compliance. Despite these challenges, organizations must ensure compliance in order to minimize the risk of security breaches and protect sensitive data. This requires a comprehensive approach involving continuous monitoring and assessment, automated testing and remediation, and regular updates to policies and procedures.

In light of these challenges, it is crucial to prioritize continuous compliance in a Kubernetes environment and invest in adopting solutions that can help automate the compliance process. [ARMO Platform](#), powered by [Kubescape](#), is a Kubernetes-focused, comprehensive solution to manage compliance in any Kubernetes environment. [Sign up today](#) to secure your Kubernetes environment and stay ahead of regulatory requirements.



Compliance Score: how ARMO simplifies compliance assessment

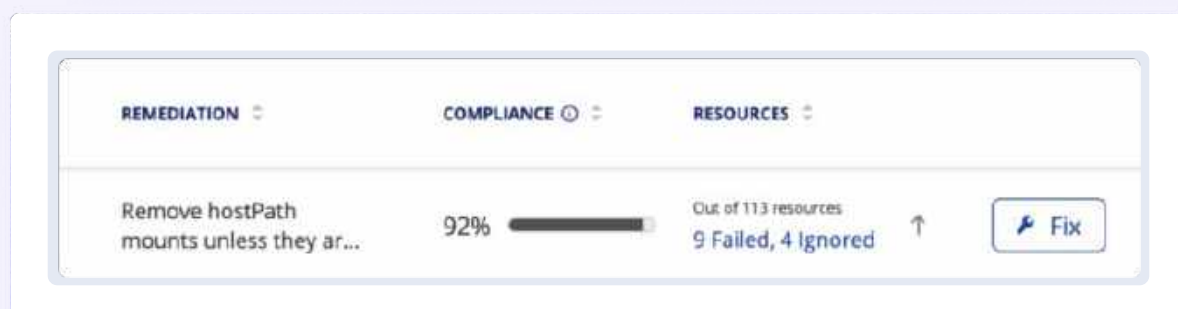
In today's digital landscape, [compliance with industry frameworks](#) is vital for businesses, Kubernetes environments are no exception. That being said, Risk Score is an illusive term. It is inconsistent between scanners and is ultimately hard to explain to stakeholders.

Compliance Score, available on ARMO Platform, offers a user-friendly method to assess compliance levels. It measures control-specific compliance and overall framework compliance. The use of a percentage metric, together with the different levels of detail, provides valuable insights that help organizations improve their compliance efforts.

Let's explore how Compliance Score works and the benefits it brings to enhancing compliance within [Kubernetes deployments](#).

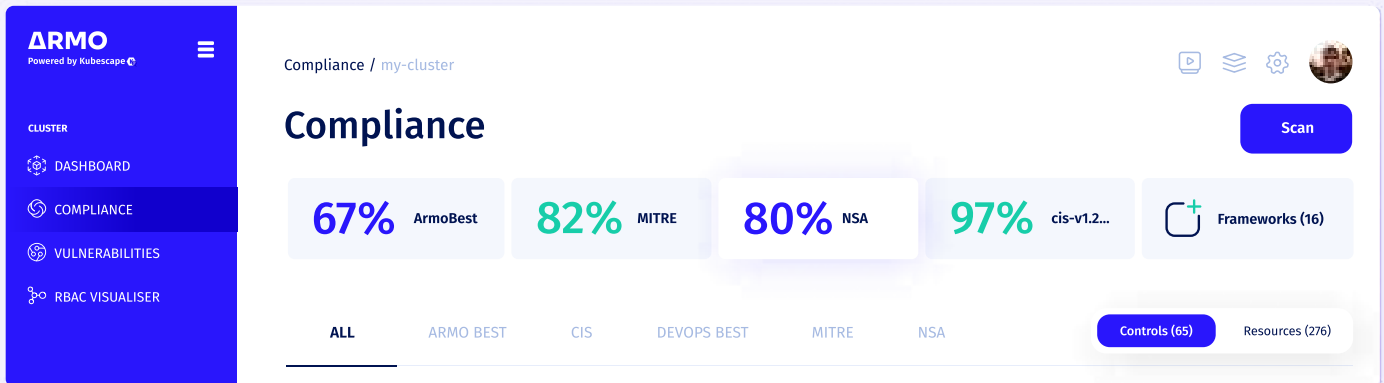
Control Compliance Score: a detailed assessment

A security control is the test run against a specific guideline in a compliance framework, to ensure that the code follows a detailed framework guideline. The Control Compliance Score evaluates compliance with individual security controls. It calculates how well resources meet the requirements of each control. This enables organizations to focus their efforts on improving control-specific compliance.



Framework Compliance Score: a holistic view

The Framework Compliance Score provides an overall assessment of compliance with a specific framework. It's calculated by averaging the Control Compliance Scores of all controls within the framework. It provides a single, easy-to-understand metric that represents the organization's compliance status.



The Compliance Score simplifies compliance assessment and empowers organizations to take targeted actions for improvement. By understanding control-specific compliance and framework-level performance, businesses can prioritize efforts, address vulnerabilities, and enhance their overall compliance posture.

Key takeaways

ARMO Platform supports the most widely used compliance frameworks: CIS, MITRE ATT&CK and NSA CISA. Folded into these frameworks, you can find the largest library of security controls on the market today (over 200 controls at the time of writing). Adding a meaningful Compliance Score to the mix provides you with the [Kubernetes security tool](#) that helps you strengthen the compliance and security for your organization.

[Try ARMO Platform today!](#) Take the guesswork out of risk scoring with Compliance Score and bridge that gap between the policy makers and those that execute it!

Achieving compliance with cloud-managed Kubernetes

Kubernetes has become a vital component in cloud-native infrastructure, enabling organizations to deploy and manage containerized applications at scale. However, compliance is crucial to modern infrastructure, especially for businesses that handle sensitive data. Organizations that adopt Kubernetes must thus also be sure to maintain the security of their infrastructure, as well as address compliance requirements to meet regulatory standards.

This section will provide an overview of compliance requirements for Kubernetes, discuss how organizations can achieve compliance certification using various Kubernetes distributions, and compare the compliance status of those distributions. It will also offer up best practices and review the importance of monitoring and remediation.

Comparing the compliance status of different Kubernetes distributions

Kubernetes distributions provided by cloud providers such as Google, Amazon, and Microsoft have become increasingly popular. They all offer different tools and strategies to help organizations achieve compliance. This section will review popular Kubernetes distributions and compare their compliance status.

Google Kubernetes Engine (GKE)

Google Cloud has a strong focus on security and compliance, and GKE provides a range of built-in security features to facilitate compliance. For example, GKE supports role-based access control (RBAC), enabling organizations to limit access to Kubernetes resources based on user roles and permissions. It also operates under a shared responsibility model.



GKE also includes Kubernetes network policies. These allow organizations to define network traffic rules for their clusters to help secure communication between pods and services. Additionally, GKE integrates with Google's Cloud Security Command Center, which provides a centralized view of security and compliance risks across all Google Cloud services.

Amazon Elastic Kubernetes Service (EKS)

As part of the AWS shared responsibility model, securing the underlying infrastructure of EKS is AWS' responsibility, while customers are responsible for securing the workloads running on EKS. To help customers achieve compliance, EKS provides a range of security features and tools; these include AWS IAM for identity and access management, AWS Key Management Service (KMS) for data encryption, and AWS CloudTrail for audit logging.

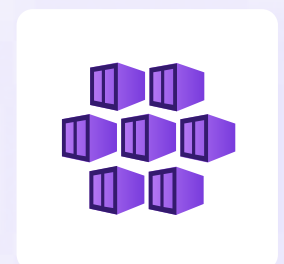


Azure Kubernetes Service (AKS)

Like EKS, AKS has a shared responsibility model. Microsoft Azure is responsible for securing the underlying infrastructure of the service, while customers take on the responsibility of securing the workloads running on AKS.

To help customers achieve compliance, AKS provides various security features such as Azure Active Directory (AD) for identity and access management, Azure Disk Encryption for data encryption, and Azure Monitor for audit logging. AKS also provides integration with third-party compliance automation tools such as Aqua Security and Sysdig.

A comprehensive set of tools, services, and certifications of the distributions is summarized in the following table.



Compliance Category



Compliance Category	Google Kubernetes Engine (GKE)	Amazon Elastic Kubernetes Service (EKS)	Azure Kubernetes Service (AKS)
Certifications	SOC 1, SOC 2, SOC 3, PCI DSS, HIPAA, ISO 27001	SOC 1, SOC 2, SOC 3, PCI DSS, HIPAA, ISO 27001	SOC 1, SOC 2, SOC 3, PCI DSS, HIPAA, ISO 27001
Security controls	Kubernetes security features, Binary Authorization, Config Connector	AWS security features, AWS IAM roles for Kubernetes, Amazon EKS best practices	Azure Security Center, Azure Policies, Azure AD for Kubernetes
Compliance automation	Google Cloud Asset Inventory, Google Cloud Security Command Center	AWS Config, AWS CloudTrail, AWS CloudWatch Events	Azure Security Center, Azure Policies, Azure Security and Compliance Blueprint
Monitoring and logging	Google Cloud operations, Cloud audit log	Amazon CloudWatch, AWS CloudTrail, AWS CloudTrail Insights	Azure Monitor, Azure Log Analytics
Certificate management	Google Certificate Manager, Let's Encrypt	AWS Certificate Manager	Azure Key Vault
Encryption	Google Cloud KMS, Google Cloud HSM	AWS KMS, AWS CloudHSM	Azure Key Vault, Azure Disk Encryption
Identity and access management	Google Cloud IAM, Google Cloud Identity-Aware Proxy	AWS IAM, AWS KMS, AWS Cognito	Azure Active Directory, Azure Kubernetes Service managed identities
Industry-specific regulations	NIST, FedRAMP, HIPAA, FISMA, CJIS	HIPAA, HITRUST, DoD, FIPS, ITAR	HIPAA, FedRAMP, DoD, DISA, CJIS

Table 1: Compliance comparison of Kubernetes providers

GKE, EKS, and AKS provide a range of security features and tools to help different organizations achieve compliance. However, it is important to note that ultimately the organization is ultimately responsible for ensuring that its workloads and applications are fully compliant with industry-specific regulations.

This may require additional measures beyond those provided by the Kubernetes distributions; these may include implementing security policies, conducting regular security audits, and establishing incident response plans. The following section will discuss best practices for achieving compliance with Kubernetes, regardless of the distribution used.

Achieving compliance with different Kubernetes distributions: best practices

As mentioned earlier, attaining a state of compliance with Kubernetes requires a combination of best practices and the use of appropriate tools and strategies, which we discuss in this section.

_Regular security assessments, vulnerability, scans and misconfigurations

Organizations should regularly perform security assessments and vulnerability scans to identify potential security risks, vulnerabilities, and misconfigurations in their Kubernetes clusters.

_Role-Based Access Control (RBAC) and audit logging

Implementing [role-based access control](#) and audit logging is critical to controlling access to Kubernetes resources and tracking changes to Kubernetes clusters.

_Adhering to industry-specific regulations

Different industries have different regulatory requirements that must be met for compliance purposes. Organizations should understand the specific regulations that apply to their industry and ensure that their Kubernetes clusters meet these requirements.

_Encrypting data in transit and at rest

Encryption of data at rest and in transit is essential for protecting sensitive data; organizations should implement encryption for all data transmitted over their Kubernetes clusters.

_Establishing a robust incident response plan

Organizations should establish a robust incident response plan that outlines the steps to be taken in the event of a security incident. This plan should include procedures for identifying, containing, and responding to security incidents.

Different Kubernetes distributions provide various tools and strategies to help organizations implement these best practices. However, Kubernetes is a dynamic environment where applications, infrastructure, and configurations change constantly. Therefore, it is critical you have continuous monitoring and automated remediation to achieve compliance with Kubernetes.

Continuous scanning, monitoring, and remediation for Kubernetes compliance

Implementing security controls in Kubernetes is not a one-time task but an ongoing process requiring continuous scanning, monitoring, and remediation. Continuous scanning helps organizations identify security risks and vulnerabilities in their Kubernetes clusters, while automated remediation can help to address these risks quickly and efficiently.

Organizations can implement several security controls in Kubernetes, such as:

Network segmentation: Divides a network into smaller subnetworks, which can help isolate and contain security incidents

Runtime protection: Monitors and controls the behavior of running applications to prevent unauthorized access or data exfiltration

Threat detection: Monitors the network for signs of malicious activity or abnormal behavior

However, implementing these security controls manually can be time-consuming and error-prone. This is one of the main reasons organizations adopt compliance automation tools to help manage their Kubernetes clusters and ensure they are meeting compliance requirements. These tools continuously monitor Kubernetes clusters to detect misconfigurations, vulnerabilities, and security incidents in real time. They can then provide automated remediation and generate compliance reports to demonstrate compliance with regulatory requirements.

In the next section, we will discuss ARMO Platform, an open-source-based [Kubernetes security tool](#) that provides advanced security features to help organizations achieve compliance with Kubernetes.

ARMO Platform

[ARMO Platform](#) is the enterprise solution based on [Kubescape](#) - A multi-cloud Kubernetes and [CI/CD security](#) single pane of glass, which is totally cloud agnostic.

ARMO Platform offers a comprehensive set of advanced security features to help organizations monitor and secure their Kubernetes clusters:

Risk analysis

Security compliance

Role-based access control (RBAC) visualization

Vulnerability scanning

Misconfiguration scanning and assisted remediation

ARMO Platform supports multiple compliance frameworks out of the box, with the largest library of controls currently available (over 200). You can create your own frameworks based on these or even [create your own controls](#).

Achieving compliance with Kubernetes is an ongoing process that requires continuous monitoring, remediation, and automation. Organizations can achieve compliance while using different Kubernetes distributions via tools and best practices. However, they need to understand the compliance status of each distribution and how it may affect their organization. To see how ARMO can help your organization achieve compliance with Kubernetes, [try it out for free today!](#)

The screenshot displays the ARMO compliance dashboard for a cluster named 'ca-terraform-eks-prod'. The dashboard provides a comprehensive overview of compliance status across several frameworks:

- AllControls:** 72% compliance
- MITRE:** 69% Avg Compliance Score (Last scan: Dec 24, 2023, 12:18)
- NSA:** 75% compliance
- SOC2:** 73% compliance
- cis-eks-t1...:** 69% compliance
- cis-v1.23-L...:** 81% compliance

Below the summary, users can compare frameworks (SOC2, cis-v1.23-t1.0.1) and view a list of failed controls. The table below details the status and remediation for these controls:

STATUS	SEVERITY	ID	CONTROL NAME	REMEDIATION	COMPLIANCE	RESOURCES
Failed	High	C-0015	List Kubernetes secrets	Monitor and approve list of users, groups and service accounts that can access secrets. Use exception mechanism to prevent repetitiv...	83%	Out of 175 resources 30 Failed, 17 Accepted
Failed	High	C-0048	HostPath mount	Remove hostPath mounts unless they are absolutely necessary and use exception mechanism to remove notifications.	98%	Out of 127 resources 2 Failed, 9 Accepted
Failed	Medium	C-0066	Secret/etcd encryption enabled	Turn on the etcd encryption in your cluster, for more see the vendor documentation.	0%	Out of 1 resource 1 Failed, 0 Accepted
Failed	Medium	C-0053	Access container service account	Verify that RBAC is enabled. Follow the least privilege principle and	57%	Out of 121 resources

Kubernetes compliance under SOC 2

System Organization Controls (SOC 2) is an auditing procedure developed by the American Institute of Certified Public Accountants (AICPA) to ensure service providers manage data securely. SOC 2 applies to all companies that process, store, or transmit client information in the cloud, and the cost of non-compliance can be significant in terms of penalties, legal action, or damage to a company's reputation.

It is therefore important for businesses that handle sensitive data to understand what they must do to meet various criteria in their data security and management practices.

In this section, we will discuss the five trust services criteria (TSC) of SOC 2 and how they apply to [Kubernetes compliance](#). We will also explore how to leverage Kubernetes security features and tools to meet SOC 2 requirements, as well as implement continuous cluster monitoring best practices and automated audit scanning tools to maintain compliance.

Understanding SOC 2 compliance for Kubernetes clusters

SOC 2 compliance requirements are divided into five trust services criteria (TSCs):

Security: Security of customer data, including access control, data encryption, and vulnerability management; **the only mandatory requirement for SOC 2 compliance**

Availability: Availability of customer data; disaster recovery and business continuity planning

Processing integrity: Accuracy and completeness of customer data; data validation and change management

Confidentiality: Protection of customer data from unauthorized disclosure, including via access control and data encryption

Privacy: Protection of customer data from unauthorized use, such as through the use of data retention and disposal policies

Kubernetes has security features such as [network policies](#), [labels](#), and [namespaces](#) for securing cluster workloads. However, there are challenges in [securing Kubernetes clusters](#) for SOC 2 compliance.

One of the biggest obstacles is the highly dynamic and ephemeral nature of containerized workloads, which makes it difficult to implement and manage security controls on network resources. Kubernetes clusters may spin up and shut down pods and services very rapidly, posing a challenge to maintaining a current network diagram for the cluster. This is because pod IP addresses change upon re-creation in the cluster and network reassignment is needed to ensure that network resources using the pods as endpoints remain in service.

This is particularly important for SOC 2-compliant workloads, where it is critical to ensure proper segmentation so that environments with sensitive data or personally identifiable information (PII) are separated from other network services. This division will protect you against a breach in the case of privilege escalation attacks.

Kubernetes security features and tools for meeting SOC 2 requirements

Kubernetes has several built-in features and design patterns that are useful for managing and securing SOC 2-compliant workloads. For one, the containerization of applications in Kubernetes allows for workload isolation, making it easier to ensure that sensitive customer data is protected.

However, Kubernetes includes a variety of other features, which we discuss below. These allow for fine-grained access control of cluster resources, thus enhancing the security of customer data in your network.

_Network segmentation of publicly accessible and internal components

Network segmentation of publicly accessible and internal components involves dividing the network into isolated segments for better security and manageability. In the context of Kubernetes, this is achieved through the use of [Ingress](#) and ClusterIPs.

The Kubernetes Ingress API object allows the administrator to manage access by external parties to the network's cluster services based on the IP address, hostname, or URL path of the incoming request. This lets organizations make sure only authorized users are granted permissions, reducing the risk of a security incident.

_Firewalling of ingress and egress traffic between applications

Organizations must control traffic flow at the IP address or port level to specify how an application is allowed to communicate with other applications within the network. In the context of Kubernetes, firewalling is achieved through the use of network policies.

Kubernetes network policies define the traffic rules for pods in a cluster; these allow the cluster administrator to control pod-to-pod network communication. They can also limit network access between pods based on their namespaces, label selectors, or IP blocks. In this way, organizations restrict communication to authorized pods only, thus decreasing the threat of a breach.

A container network interface (CNI) is necessary when implementing Kubernetes network policies. By applying these to a pod, traffic policy can be switched from default-allow to default-deny, where only allow-listed traffic is permitted. This enables fine-grained control of network traffic between applications, ensuring that only authorized traffic is granted access and improving the security of the overall system.

_Encryption of data in transit

Encryption of data in transit is an important requirement for service providers, as sensitive data and PII are often processed and transmitted over internal and public networks in the cloud.

For data transmission within the cluster, it is recommended to use a [service mesh](#) such as Istio to encrypt all traffic between pods and secure communication between services. The Istio service mesh provides the ability to enforce [TLS](#) for encryption between services within the mesh. For encryption of transmission over public networks, [cert-manager](#) automates the management and issuance of certificates in the cluster to secure network resources, such as Ingress resources and the Istio service mesh.

_Enforce workload isolation to limit blast radius

Implementing security controls to prevent unauthorized access and other vulnerabilities is a critical security requirement in SOC 2 compliance, as the consequences resulting from an uncontained data breach can be severe. Organizations must limit the blast radius of security breaches by applying the principle of least privilege; this will ensure that only authorized users have access to network resources utilized by workloads handling sensitive data.

Identity-aware microsegmentation enables DevSecOps teams to enforce workload isolation in clusters running multiple applications on the same cluster; they achieve this by securing the workloads individually with workload-specific policies and access controls. In Kubernetes, identity-aware microsegmentation can be implemented using the following approaches:

- **Role-Based Access Control (RBAC)** restricts permissions to resources based on users' roles.
- **Kubernetes namespaces** provide a mechanism for [scoping a cluster](#) into isolated sub-clusters with their own groups of cluster resources.
- **Kubernetes labels** apply network policies to a [subset of cluster](#) resources within the same namespace.
- **Lightweight Directory Access Protocol (LDAP) using OpenID Connect (OIDC)** providers manages user authentication and authorization so that access to sensitive data is restricted to authorized users only.

Guidelines for ensuring SOC 2 compliance over time

As SOC 2 audits are typically conducted annually, organizations must establish robust monitoring practices, which we discuss in the following sections. These will ensure that your security controls and operations continue to comply with SOC 2 requirements over time. For Kubernetes compliance under SOC 2, such practices are aided by the use of scanning and policy enforcement tools to identify security vulnerabilities and misconfigurations in the cluster.

_Continuous logging, monitoring, and automated audit scanning

Setting up continuous logging, monitoring, and automated audit scanning is critical to demonstrate and maintain continuous SOC 2 compliance, as well as mitigate any active threats. For Kubernetes clusters, continuous monitoring should be implemented at each layer in the cluster infrastructure stack, which includes container images in pods and network communications involving cluster resources.

Scanning container images enables you to detect security vulnerabilities inherited from external dependencies, such as open-source packages and third-party registries.

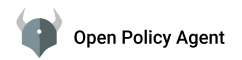
_Preventing misconfigurations

It is important to implement strict security policies to prevent any misconfigurations. [Kubernetes' Pod Security Standards](#) and Pod Security Admission enforce policies to make sure pods are deployed with the necessary security contexts and capabilities. Alternatively, CNCF projects such as [Kyverno](#) explicitly list required capabilities and enforce security policies across the cluster.



[Open Policy Agent \(OPA\)](#) works to identify [misconfigurations](#) such as the use of default passwords. This is particularly important in Kubernetes environments managed by DevSecOps teams, where misconfigurations may be inadvertently introduced via automated deployment tools in a [CI/CD pipeline](#). OPA is a policy engine that integrates with Kubernetes to provide fine-grained control over resource access and configuration.

By leveraging OPA, DevSecOps teams can ensure that all configurations and policies adhere to compliance standards such as SOC 2.



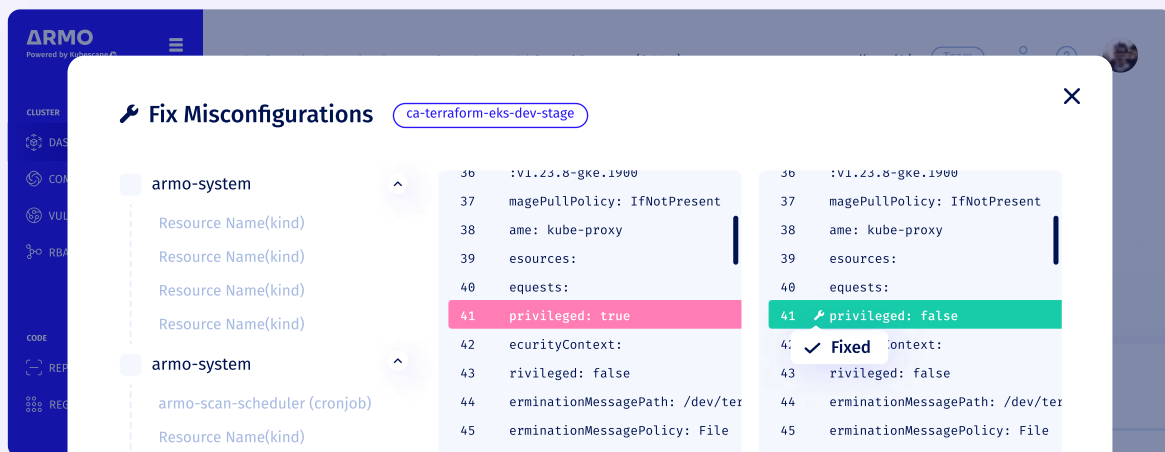
_Traffic logging

Logging traffic to monitor access to network resources and sensitive customer data in the cluster is key to ensuring data is being managed properly. FluentD, Prometheus, and Grafana are popular tools for collecting and visualizing logs from the Kubernetes cluster; these allow DevSecOps engineers to monitor for all instances of access being granted to network resources, pods, and storage.



_Automated audit scanning

Automated audit scans are essential for ensuring that Kubernetes clusters are SOC 2 compliant. Scanning the Kubernetes cluster against [Kubernetes security frameworks](#) such as the [CIS Benchmark](#) can help you identify and remediate security misconfigurations. This, in turn, assists organizations in preventing breaches involving sensitive customer data, which can result in significant penalties and loss of customer trust.



Key Takeaways

Ensuring [SOC 2 compliance](#) is crucial for companies that handle sensitive client information, as the cost of non-compliance can be significant in terms of financial penalties and damage to reputation. DevSecOps engineers can adhere to the needed criteria by leveraging existing [Kubernetes security tools](#) and following best practices for cluster security.

[ARMO Platform](#) powered by [Kubescape](#) gives you over 200 security controls out-of-the-box, based on recognized industry frameworks, as well as RBAC visualization. All of which help you achieve compliance. The dashboards and notifications that monitor drift, enable you to work towards ensuring compliance over time.



Kubernetes compliance under GDPR

The General Data Protection Regulation (GDPR) is a data privacy and security regulation in the European Union (EU) that aims to protect individuals' personal data collected and processed by businesses.

The financial penalties for a company that is found to be non-compliant with GDPR can be significant: €20 million or 4% of its [annual global revenues](#). Hence, it is essential for small and medium-sized businesses (SMBs) to understand how the regulation applies to their data processing and management practices.

In this section, we will discuss the GDPR principles and requirements on data privacy, how GDPR applies to [Kubernetes compliance](#) and best practices for achieving compliance in Kubernetes clusters.

GDPR principles and requirements on data privacy

Organizations must abide by seven fundamental principles listed in [Section 5 of GDPR](#) when handling personal data:

- Lawfulness, fairness, and transparency
- Purpose limitation
- Data minimization
- Accuracy
- Storage limitation
- Integrity and confidentiality
- Accountability

In particular, GDPR requires companies to implement “[appropriate technical and organizational measures](#)” to ensure the security of personal data. This includes data protection “[by design and by default](#)” meaning that privacy considerations must be built into the architecture and operation of systems that process personal data.

GDPR additionally grants EU-based individuals the right to access their personal data and to have their personal data erased. Companies must also notify the relevant authorities within 72 hours of a personal data breach.

How GDPR applies to Kubernetes compliance

For SMBs that are using Kubernetes, compliance with GDPR requirements for data privacy can be confusing, as GDPR does not specify how the principles should be applied in practice.

Leveraging [existing Kubernetes security frameworks and guidance](#) is key. The Kubernetes community has developed several security frameworks that provide a comprehensive set of guidelines for securing Kubernetes resources, including network and data management, as well as access control and authentication.

There are some primary considerations for Kubernetes compliance under GDPR, a few of which we discuss below.

[_Access control and authentication](#)

When granting access to resources in the Kubernetes cluster, apply the principle of least privilege so that only authorized users have access to sensitive data. User authentication via mechanisms such as OpenID Connect (OIDC) and client certificates are also supported in Kubernetes so that all API requests to the [Kubernetes API server](#) in the cluster are authenticated.

[_Data management at rest and in transit](#)

Credentials and sensitive data should be encrypted at rest, while data in transit over the network should be secured by configuring control plane components to use encrypted communication via [Transport Layer Security \(TLS\)](#).

_Network segmentation

Leaving all traffic open within a network, that is, allowing all inbound and outbound traffic between pods and clusters, could lead to a compromised cluster when another cluster in the network is affected. Only necessary traffic should be allowed; all else should be denied by default if it is not required for the workloads to guarantee that only authorized pods can communicate with each other.

_Auditing and continuous monitoring

Enabling audit logging and continuous monitoring in the Kubernetes cluster is necessary for companies to provide evidence of having implemented “appropriate technical and organisational measures” in case of a personal data breach.

Best practices for Kubernetes compliance under GDPR

The following are some security best practices that DevSecOps engineers in SMBs can apply in their Kubernetes environments for GDPR compliance.

_Set up Role-Based Access Control

Role-based access control (RBAC) is a built-in security model in Kubernetes for controlling access to resources based on roles and permissions. It is a best practice to set up RBAC for Kubernetes clusters to implement the principle of least privilege and ensure that only authorized users have access to network resources. This, in turn, helps prevent data breaches.

Organizations can implement RBAC in Kubernetes through the creation of roles, role bindings, and service accounts. Roles define the permissions that users or service accounts have within the cluster. Service accounts are typically used by pods inside the cluster to perform certain actions on cluster resources; they can be used to grant permissions to applications or services running in the cluster without the need for a client certificate (which is required for users). Role bindings map roles to users or service accounts.

You will first need to define the roles and permissions for the cluster. Next, create service accounts for the applications and services that will need to be able to access the cluster. Finally, you can map the roles to the service accounts using role bindings.

It is recommended that DevSecOps teams restrict access to the Kubernetes cluster to the minimum necessary for maintaining operations. This can be accomplished by creating different roles for different tasks, such as view-only access for monitoring and reporting and full administrative access for managing the cluster.

_Use encryption to secure data at rest and in transit

To ensure data privacy and security, it is important to encrypt data at rest, such as when it is stored in the Kubernetes cluster. One way to achieve this is by using encrypted [Kubernetes Secrets](#) and encrypted storage volumes. Kubernetes Secrets allow you to store sensitive information, such as passwords and keys, securely in the cluster, while encrypted storage volumes provide an encrypted storage solution for your data.

Encryption is also important when data is in transit, for instance, over the network. It is recommended to use a [service mesh](#) such as Istio to encrypt all traffic between pods and to secure communication between services within the cluster.

_Implement network segmentation and firewalling

Network segmentation involves dividing the network into isolated segments for better security and manageability. In the context of Kubernetes, network segmentation is achieved through the use of network policies.

[Kubernetes network policies](#) define the traffic rules for pods in a cluster, which allows the cluster administrator to control pod-to-pod network communications. Companies can also leverage network policies to limit network access between pods based on their [namespaces](#), label selectors, or IP blocks. This helps make sure that only authorized pods can communicate with each other, thus lowering the risk of data breaches and unauthorized access.

In addition to network policies, [Ingress](#) is another important component of network segmentation in Kubernetes. Ingress allows the administrator to manage external network access to cluster services based on the IP address, host name, or URL path of the incoming request. This way, only authorized users can access the cluster and its services, thereby reducing the risk of a breach.

_Setting up continuous logging, monitoring, and alerting

Setting up continuous logging, monitoring, and alerting is critical to demonstrate compliance with GDPR requirements. This can be done by implementing data loss prevention (DLP) systems; cluster logging, monitoring, and alerting; and continuous compliance monitoring. DLP systems use deep packet inspection (DPI) to inspect all network connections. This is important because it enables the detection of any unauthorized access to personal data. In a Kubernetes environment, DLP systems can be integrated into the cluster as a custom resource, allowing them to inspect all traffic flow in the cluster.

Cluster logging, monitoring, and alerting are also critical to making sure that Kubernetes clusters are GDPR-compliant. FluentD, Prometheus, and Grafana are popular tools for collecting and visualizing logs from the Kubernetes cluster, allowing DevSecOps engineers to monitor the cluster for security incidents.

Continuous compliance monitoring is another essential factor in being GDPR-compliant. Automated scanning of the Kubernetes cluster against Kubernetes security frameworks such as [CIS Benchmark](#) and PCI DSS can help to identify and remediate security misconfigurations. This, in turn, enables you to avoid data breaches, which can result in significant fines under GDPR. Being able to demonstrate that the cluster is GDPR-compliant is also important in the event of an audit.

Key Takeaways

Ensuring compliance with GDPR is a must for SMBs using Kubernetes, as the penalties for non-compliance can be significant. By leveraging existing Kubernetes security frameworks to implement best practices for cluster security and log auditing, DevSecOps engineers can design and operate their Kubernetes clusters in a manner that meets GDPR requirements and protects the privacy and security of personal data.

Kubernetes compliance under HIPAA

To protect your organization from security breaches and data leaks, it is increasingly important to ensure that your [Kubernetes cluster is secure](#) and compliant with legal and industry [best practices](#).

[Compliance](#) is especially important in regulated industries. To achieve it, development and IT operations teams should have defined roles and tasks, along with tools and practices that include security frameworks and policies, API server monitoring, and Kubernetes audit logs.

This section will also cover popular open-source Kubernetes compliance tools like Open Policy Agent (OPA) and Kubescape.

Security guidance framework — HIPAA

Security guidance frameworks (SGFs) help secure containerized applications and services. Examples of SGFs include the Health Insurance Portability and Accountability Act (HIPAA) in the US, [GDPR](#) in the EU, and PCI DSS. These frameworks are designed to protect sensitive personal data.

HIPAA rules apply to medical service providers, health plans, research facilities, and insurance companies that work with patient data. The requirement to secure protected health information (PHI) also applies to business associates.

The HIPAA privacy rule

The Privacy Rule gives individuals full control over their PHI. HIPAA-regulated organizations must get consent before processing PHI. Regulated entities are responsible for privacy violations that result in unauthorized disclosure of PHI.

_Seven principles for HIPAA compliance

Weighing the risks of migrating data, i.e. security breaches, data losses, vendor lock-in

Establishing a company-wide security culture (from DevOps to DevSecOps)

Maintaining separate development and testing environments with careful access controls

Securing data by using container images only from trusted and reputable repositories to avoid malware contamination; using only current images from allow-listed, reputable repositories is crucial for this reason.

Ensuring regular vulnerability scanning and monitoring at every level of deployment

Protecting (encrypting) data entering and leaving your containers

Regular, automatic backups of user data

Containers provide high availability; however, that alone is insufficient. Only by replicating your environment's images, associated databases, deployments, persistent volumes for pods, and resources can you guarantee access to your environment in the event of failure.

_HIPAA compliance strategies

To ensure HIPAA compliance, use secure technologies, follow configuration guidelines, keep secrets outside of images, encrypt data in transit and at rest, and encrypt all PHI transmissions over the Kubernetes network.

Additional important steps include:

Using a secure connection when pushing to or pulling from the registry

Ensuring that the container uses the most recent image release

Segmenting network traffic

Following container runtime configuration standards like the [Center for Internet Security \(CIS\) benchmarks](#)

Identifying potential breaches at the infrastructure and container levels, using a container-specific operating system

Ensuring that containers operate with the fewest permissions necessary

Generally speaking, organizations are required to use trusted CDNs and verified cloud storage services (for example, Amazon ECR and JFrog Artifactory) for achieving image integrity and authenticity at every stage of the image life cycle, from creation to deployment.

Privacy protection frontier with internals in mind

Regulated companies using Kubernetes must install an intrusion detection system (IDS) like Falco to comply with Privacy Rules. Falco uses Linux kernel features to detect any unusual actions in production pods that may indicate a leak of PHI.

To prevent these leaks, it is recommended to securely log all important boundary events, such as logins, logouts, API calls, and database requests. This can be done by hosting Elasticsearch on a separate Kubernetes cluster and sending live application logs to it using Fluentd. Additionally, [Kubernetes API](#) logs should be sent to the same tamper-proof logging environment as the application logs to ensure their security even if the production environment is compromised.

RBAC and policy enforcement

A key aspect of Kubernetes security is [role-based access control \(RBAC\)](#), which allows control over access to different resources in the cluster. Next, we will discuss RBAC and policy enforcement, as well as the use of policy engines, specifically Open Policy Agent (OPA), in Kubernetes.

Policy engines: Advantages and drawbacks

Policy engines are software tools that help automate policy enforcement by allowing you to define policies in a declarative way, making it easier to manage policies across different applications and environments. They offer several advantages, such as:

Consistency in enforcing policies across different applications and environments

Automation that reduces the risk of human error

Granularity in enforcement that provides fine-grained control over access to resources

Flexibility that allows policies to be easily updated or modified

However, there are also some drawbacks to using policy engines. One key challenge is defining policies that are both comprehensive and easy to manage.

OPA-based techniques for policy enforcement

Open Policy Agent is an open-source policy engine for Kubernetes. It provides a flexible and declarative way of defining policies for multiple applications and environments.

OPA is built around a simple idea: defining **policies as code (PaC)**. This approach involves using a programming language or declarative syntax and storing them in a version control system, just like code.

OPA enforces these policies at runtime, making it easier to manage them across different environments and ensure consistent and automatic enforcement.

OPA has several **advantages** over other policy engines. For example, its policies can be easily tested and debugged, and it provides a rich set of APIs for integrating with different applications and environments.

However, OPA also has its **drawbacks**, including a steep learning curve for its proprietary language Rego, lack of GUI, complexity and performance issues. Misconfigurations or vulnerabilities in OPA could potentially allow attackers to bypass policies.

Pod security standards

There are a [set of guidelines](#) that are designed to enhance the security of Kubernetes pods, which are the smallest deployable units in the Kubernetes platform. Following PSS can enforce compliance with HIPAA, PCI DSS, and GDPR.

Using OPA with PSS

When using OPA with PSS, organizations can define policies that follow PSS guidelines, such as ensuring that pods run as non-root users, restricting the use of privileged containers, or enforcing network policies. OPA can help automate the enforcement of these policies across a Kubernetes cluster, and it can provide real-time feedback on policy violations or compliance status.

At the same time, OPA and PSS can be overwhelming to configure, especially for large and complex environments. It is important to have a clear understanding of the access control requirements and carefully design and test policies before deployment in production.

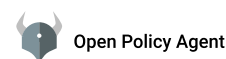
Kyverno's policy-as-code approach

To protect the cluster, you can also refer to the least privileged Pod permission model, a part of the Kubernetes Pod Security Standards. The model's principle states that Kubernetes pods should be granted only the permissions necessary to perform their intended functions. This helps reduce the risk of unauthorized access or malicious activity by restricting the capabilities of pods and containers.

Permissions may be implemented like PaC, and Kyverno is a powerful tool for implementing the PaC approach in Kubernetes.

Both Kyverno and OPA support Kubernetes Pod Security Standards.

Kyverno is a potential alternative to OPA and its complex technical needs. It offers a structure similar to Kubernetes in terms of object description and reconciliation, thanks to its declarative policy expression and Kubernetes-specific architecture. Policy elaboration is greatly simplified by Kyverno.



Unfortunately, Kyverno also has disadvantages:

It lacks a programming language for defining policies. Instead, it uses YAML files, which can become cumbersome.

It offers less flexibility in defining policies. Kyverno [may not be able to enforce policies on certain types of Kubernetes resources](#), such as custom resources. Meanwhile, OPA can be used with any Kubernetes resource, including custom resources.

The PaC approach requires a certain level of programming and technical expertise. Users should be comfortable with YAML and other declarative syntaxes and have a basic understanding of programming concepts such as loops, conditionals, and functions.

ARMO Platform

[ARMO Platform](#), powered by [Kubescape](#) a CNCF sandbox project, is a powerful tool designed to comprehensively test the security of Kubernetes infrastructure. It scans for potential vulnerabilities and compliance violations across various Kubernetes objects, including pods, services, and deployments. It provides in-depth analysis of identified security weaknesses, misconfigurations, and compliance violations, allowing for quick and effective remediation.

ARMO Platform uses OPA behind the scenes and has the [largest library of controls in the industry](#), with over 200 controls that are based on CIS, NSA, and MITRE ATT&CK. As a result, users of Kubescape do not need to write policies. Thanks to all of that, ARMO Platform users can easily identify potential security weaknesses and compliance violations. In addition, they can boost their effectiveness by leveraging automatic remediation. By that they can cleverly ensure their Kubernetes clusters are secure and compliant.

Key Takeaways

Compliance is an essential aspect of Kubernetes security. Through the right practices, companies can ensure that their Kubernetes clusters remain secure and compliant with major regulations.

This involves starting with a security framework - ensuring the quality and integrity of the software images to be deployed, using RBAC, enforcing PSSs, monitoring the Kubernetes API server, regularly reviewing and updating policies, and using Kubernetes audit logs. By following these guidelines and leveraging narrow profile tools, you can stay ahead of compliance issues and ensure the security of containerized applications and services.

ARMO Platform and Kubescape help users become HIPAA-compliant by mastering security checks and applied configurations. By ensuring secure and proven resources are used in deploying apps, ARMO's solution is a valuable tool for identifying and preventing security issues.



Kubernetes compliance under ISO 27001

What is ISO 27001?

ISO 27001 is an international security standard that offers organizations a systematic approach to handling sensitive data. Its guidelines help organizations of all sizes and types identify and manage information security risks, as well as ensure the confidentiality, integrity, and availability of their data assets.

While organizations have to meet certain requirements to receive ISO 27001 certification, the framework is adaptable to a company's specific needs.

ISO 27001 requires a risk-based approach to security. This entails identifying and assessing threats to data assets and then implementing controls to counter those risks. Organizations can thus prioritize their security efforts and allocate resources effectively.

The benefits of ISO 27001 compliance are many. It can improve risk management; protect against cyber threats; and demonstrate to clients, partners, and regulators a commitment to data security. Adhering to the standard also helps address weaknesses in security practices.

Kubernetes: sensitive workloads and attack surface

Kubernetes provides some powerful features for managing and securing sensitive workloads, such as [role-based access control \(RBAC\)](#), [network policies](#), and [container security contexts](#). However, it is still essential for companies to harden and secure their Kubernetes environment itself. This involves implementing best practices for securing the [Kubernetes control plane](#), including limiting access to the [Kubernetes API server](#), using secure network communication protocols, and monitoring Kubernetes clusters for breaches.

One crucial consideration is the attack surface. As a complex distributed system, Kubernetes presents numerous potential attack vectors that must be addressed to comply with ISO 27001. These may be [misconfigured Kubernetes resources](#), insecure container images, or vulnerabilities in the underlying infrastructure.

By taking a proactive approach to Kubernetes security, businesses can ensure they are able to meet the requirements of ISO 27001 to properly protect their sensitive data and applications.

ISO 27001 and Kubernetes

In this section, we will cover [the relevant clauses of ISO 27001](#), discussing the specific challenges and considerations for securing Kubernetes clusters per the ISO 27001 standard. The first three clauses - namely “Scope,” “Normative references,” and “Terms and definitions” - explain the details of the ISO 27001 standard. Thus, for the purpose of [Kubernetes compliance](#), we will start from the fourth section.

_Section 4: context of the organization

Organizations must first identify the assets they need to protect and the risks they have to manage. In this section, businesses should define the scope of their Kubernetes environment and establish information security objectives and policies. For example, the former can be determined by identifying the namespaces, clusters, and nodes that are part of the Kubernetes platform.

_Section 5: leadership

Organizations must first identify the assets they need to protect and the risks they have to manage. In this section, businesses should define the scope of their Kubernetes environment and establish information security objectives and policies. For example, the former can be determined by identifying the namespaces, clusters, and nodes that are part of the Kubernetes platform.

_Section 6: planning

The risk assessment phase should identify the assets, vulnerabilities, and threats associated with the Kubernetes environment. The risk mitigation plan should outline the controls that will be implemented to manage these risks, such as RBAC, network policies, and secure coding practices.

_Section 7: support

Businesses must provide appropriate resources, training, and awareness related to Kubernetes security; this may take the form of teaching developers and administrators [Kubernetes security best practices](#) and Kubernetes-native features. Companies should also provide resources such as tools and documentation to help developers and administrators implement security controls [in their Kubernetes clusters](#).

_Section 8: operation

Kubernetes is a relatively new technology, and there is currently a talent gap in terms of people who are skilled in its use. Implementing information security controls, such as incident management, change management, and backup and recovery processes in ephemeral and dynamic Kubernetes environments, can be challenging. Businesses must adapt to these requirements to ensure their Kubernetes clusters remain secure.

_Section 9: performance evaluation

Companies need to monitor and evaluate the security performance of their Kubernetes infrastructure. Organizations can implement this using metrics and audits to measure their security controls' effectiveness and identify areas in need of improvement.

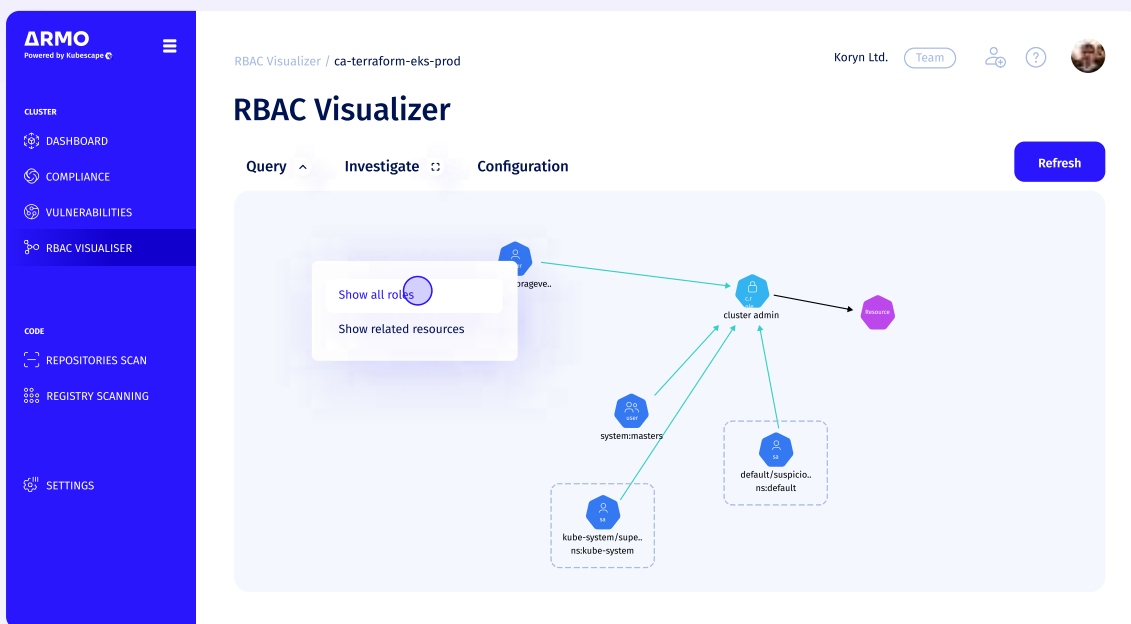
_Section 10: improvement

Continuous improvement is critical for maintaining the security of the Kubernetes landscape. Businesses must take corrective actions, adjust risk treatment plans, and provide security awareness programs.

Kubernetes security features and tools that support ISO 27001 compliance

The Kubernetes ecosystem offers several security features and tools to meet the requirements of ISO 27001:

Role-based access control (RBAC): Kubernetes RBAC policies define roles with specific permissions for accessing and managing Kubernetes resources.



Network policies: These define and enforce traffic rules between Kubernetes pods and services to limit the attack surface and prevent unauthorized access to sensitive data.

Secrets management: Kubernetes offers a built-in secrets management feature that allows administrators to securely store and manage sensitive data such as passwords, API keys, and other credentials. However, Secrets are stored unencrypted in the API server's data store. This allows anyone with API access or access to etcd to retrieve or modify them, and anyone authorized to create a Pod in a namespace can indirectly access any Secret in that namespace. Therefore, it is essential to consider [encrypting secret data at rest](#) to use secrets in a secure way.

Image scanning: The Kubernetes ecosystem offers several scanning tools to discover vulnerabilities in container images, such as [Clair](#) and [Kubescape](#).

Logging and monitoring: Kubernetes works efficiently with several logging and monitoring tools, such as [Prometheus](#) and [Fluentd](#); companies can use these to monitor Kubernetes clusters for security incidents and to collect and analyze logs.

Compliance automation: [Open Policy Agent \(OPA\)](#) and [Gatekeeper](#) automate compliance checks and ensure that Kubernetes configurations comply with ISO 27001 requirements.

By leveraging these features and tools, businesses can enhance the security of their Kubernetes environments and ensure they achieve ISO 27001 compliance. However, it is essential to note that these tools should be part of a comprehensive security strategy that includes risk assessments, security policies, and ongoing monitoring and auditing of their Kubernetes clusters.

In the next section, we will review a few best practices to help companies keep their attack surface tight.

Best practices for achieving Kubernetes compliance under ISO 27001

There are widely accepted best practices in the industry to help organizations comply with ISO 27001.

_Define a security policy

Businesses must have a clear and comprehensive security policy in place. This should detail the organization's approach to managing information security risks in its Kubernetes landscape, including the roles and responsibilities of all stakeholders.

_Implement access controls

These ensure that only authorized users gain access to sensitive data in a Kubernetes

_Implement access controls

These ensure that only authorized users gain access to sensitive data in a Kubernetes environment. Businesses should implement role-based access control (RBAC) to restrict permissions to Kubernetes resources and enforce the principle of least privilege.

_Use network security measures

Kubernetes provides a range of network security measures that organizations can use to protect sensitive data, including network policies and secure network plugins. Network policies enable businesses to define how pods communicate with each other and external resources, while secure network plugins encrypt network traffic to prevent data interception and theft.

_Employ encryption techniques

Encryption is an essential tool for protecting sensitive data in Kubernetes clusters. Via tools such as Transport Layer Security (TLS) and encrypted storage volumes, companies should encrypt their data to protect it at rest and in transit.

_Stay on top of the latest security best practices

Kubernetes is a rapidly evolving technology, so businesses must be up-to-date with recent best practices to maintain ISO 27001 compliance. This requires keeping current on emerging threats and vulnerabilities, promptly implementing patches and updates, and adopting new security measures and controls.

By following these best practices, businesses can adhere to ISO 27001 for their Kubernetes environments, ensuring that sensitive data is protected and information security risks are managed effectively.

Key Takeaways

As a widely recognized international standard for information security management, ISO 27001 provides a framework for businesses to manage data security risks and protect sensitive information. Kubernetes is also an excellent tool for managing and securing sensitive workloads and can help organizations meet the requirements of ISO 27001.

Businesses should implement best practices and utilize the features of Kubernetes and tools of the cloud-native ecosystem as much as possible. By aligning their Kubernetes cluster configuration and management with the guidelines of ISO 27001, companies can achieve compliance and protect their sensitive data from being compromised.



Kubernetes Compliance Under PCI

The [Payment Card Industry Data Security Standard \(PCI DSS\)](#), established by the Payment Card Industry Security Standards Council (PCI SSC), serves to protect cardholder data. The framework applies to all companies that process, store, or transmit such data, regardless of their size or location. As penalties may be imposed by card brands (Visa, Mastercard, etc.) or payment processors for non-compliance, it is important for businesses that handle sensitive payment information to understand how the regulation applies to their data security and management practices.

In this section, we will discuss the use of Kubernetes for securing and managing PCI-compliant workloads, the 12 requirements of PCI DSS, and how they apply to Kubernetes compliance. We will also introduce Kubernetes security features and tools for adhering to PCI DSS, along with best practices for continuous cluster monitoring and auditing to ensure your Kubernetes cluster remains PCI-compliant.

PCI DSS requirements

PCI DSS covers [12 requirements](#) that companies must meet to protect cardholder data. For the purpose of Kubernetes, the following are relevant:

Network security controls and firewall configuration: Kubernetes provides a variety of tools for securing network traffic, including network policies, service accounts, and integration with ingress controllers to deploy ingress resources. Organizations can use these tools to ensure that only authorized traffic is allowed to reach sensitive payment data.

Default passwords and security parameters: It is particularly important in Kubernetes to avoid these; this is because automated deployment tools may use default settings unless they are explicitly configured and encrypted.

Tracking and monitoring all access to network resources and cardholder data: This requires a continuous monitoring and auditing solution that can track and log all permissions granted to view and use network resources, pods, and storage within the cluster.

Regular testing of security systems and processes: Companies must implement routine scans for vulnerabilities, penetration testing, and automated testing, along with remediation, in the application deployment pipeline.

There are some challenges in securing Kubernetes clusters for PCI compliance. One of the biggest concerns is the highly dynamic and ephemeral nature of containerized workloads. Kubernetes clusters may spin up and shut down pods and services very rapidly, which can make it difficult to track and monitor access to network resources and cardholder data.

Another challenge is maintaining a current network diagram for the cluster. With so many moving parts, it can be difficult to keep an eye on all of the connections between pods, services, and external parties. This is particularly important for achieving PCI-compliant workloads, as it is critical to maintain a clear separation between the cardholder data environment and other network services.

Kubernetes security features and tools for meeting PCI DSS requirements

Kubernetes has several built-in features and design patterns that are useful in managing and securing PCI-compliant workloads. The containerization of applications in Kubernetes allows for process isolation, making it easier to ensure that sensitive payment data is protected. Kubernetes also includes a variety of features that allow for fine-grained control of access to cluster resources; this enhances the security of cardholder data in your network.

DevSecOps engineers working at companies handling sensitive payment data can apply the following security features and tools in their Kubernetes environments to meet PCI DSS requirements.

_Implementing network segmentation

Network segmentation of publicly accessible and internal components involves dividing the network into isolated segments for better security and manageability. In the context of Kubernetes, the network segmentation of publicly accessible and internal components is achieved through the use of [Ingress](#) and ClusterIPs.

The Kubernetes Ingress API object allows the administrator to manage access by external parties to the network's cluster services based on the IP address, hostname, or URL path of the incoming request. This lets organizations make sure only authorized users are granted permissions, reducing the risk of a security incident.

_Firewalling of inbound and outbound traffic between applications

Organizations must control traffic flow at the IP address or port level to specify how an application is allowed to communicate with other applications within the network. In the context of Kubernetes, firewalling is achieved through the use of network policies.

[Kubernetes network policies](#) define the traffic rules for pods in a cluster, which allows the cluster administrator to control pod-to-pod network communication. They can also limit network access between pods based on their [namespaces](#), label selectors, or IP blocks. In this way, organizations restrict communication to authorized pods only, thus decreasing the threat of a breach.

A container network interface (CNI) is necessary when implementing Kubernetes network policies. By applying these to a pod, traffic policy can be switched from default-allow to default-deny, where only allow-listed traffic is permitted. This enables fine-grained control of network traffic between applications, ensuring that only authorized traffic is granted access and improving the security of the overall system.

_Encrypting data in transit

Encryption of data in transit is a critical security requirement for PCI compliance, as cardholder data is often transmitted over internal and public networks such as payment networks.

For data transmission within the cluster, it is recommended to use a [service mesh](#) such as Istio to encrypt all traffic between pods and secure communication between services. The Istio service mesh provides the ability to enforce TLS for encryption between services within the mesh. For encryption of transmission over public networks, [cert-manager](#) automates the management and issuance of certificates in the cluster to secure network resources, such as Ingress resources and the Istio service mesh; Kubernetes also provides built-in support for provisioning TLS certificates via the certificates.k8s.io API.

_Restricting cardholder data to a Need-to-Know basis

Restricting cardholder data to being solely viewed on a need-to-know basis is a fundamental principle of PCI compliance. The principle of least privilege ensures that only authorized users have access to network resources.

Organizations can implement role-based access control (RBAC) to manage permissions for resources within the cluster, providing fine-grained control based on different users and groups. Leveraging lightweight directory access protocol (LDAP) via OpenID Connect (OIDC), providers can manage user authentication and authorization.

All of these practices will enable organizations to reduce unauthorized access to cardholder data.

Monitoring and auditing best practices for PCI compliance

_Continuous logging, monitoring, and automated audit scanning

Setting up continuous logging, monitoring, and automated audit scanning is critical to demonstrate and maintain continuous PCI compliance, as well as mitigate any active threats. For Kubernetes clusters, continuous monitoring should be implemented at each layer in the cluster infrastructure stack, which includes container images in pods and network communications involving cluster resources.

Scanning container images enables you to detect security vulnerabilities inherited from external dependencies, such as open-source packages and third-party registries.

_Preventing misconfigurations

It is important to enforce strict security policies to prevent any misconfigurations. Kubernetes' Pod Security Standards and Pod Security Admission enforce policies to make sure pods are deployed with the necessary security contexts and capabilities. Alternatively, CNCF projects such as [Kyverno](#) explicitly list required capabilities and enforce security policies across the cluster.

[Open Policy Agent \(OPA\)](#) works to identify misconfigurations such as the use of default passwords. This is particularly important in Kubernetes environments managed by DevSecOps teams, where misconfigurations may be inadvertently introduced via automated deployment tools in a CI/CD pipeline. OPA is a policy engine that integrates with Kubernetes to provide fine-grained control over resource access and configuration. By leveraging OPA, DevSecOps teams can ensure that all configurations and policies comply with regulatory standards such as PCI DSS.

_Traffic logging

Traffic logging to monitor access to network resources and cardholder data in the cluster is critical in ensuring continuous PCI compliance. FluentD, Prometheus, and Grafana are popular tools for collecting and visualizing logs from the Kubernetes cluster, allowing DevSecOps engineers to monitor for all instances of access being granted to network resources, pods, and storage.

_Automated audit scanning

Automated audit scanning is essential for ensuring that Kubernetes clusters are PCI-compliant. Scanning of the Kubernetes cluster against Kubernetes security frameworks such as the [CIS Benchmark](#) and PCI DSS can help to identify and remediate security misconfigurations. This, in turn, assists organizations to prevent breaches involving sensitive cardholder data, which can result in significant penalties from card brands or payment processors.

Key Takeaways

Ensuring compliance with PCI is critical for companies that handle sensitive payment information, as penalties may be imposed for non-compliance, not to mention the risk of a breach.

DevSecOps engineers must design and operate their Kubernetes clusters in a PCI-compliant manner to guarantee the security of network resources and cardholder data. They can achieve this by leveraging existing Kubernetes security tools and following best practices for cluster security. Organizations can also implement continuous compliance through the use of monitoring and auditing tools such as [ARMO Platform](#) powered by [Kubescape](#).

About ARMO

ARMO uniquely specializes in Kubernetes security. Our mission is to equip Security and DevOps teams with noise-free, contextual, and actionable security insights using Kubernetes-native solutions. ARMO is an open-source-driven company and the creator of Kubescape, one of the most complete and fastest growing Kubernetes security open-source project, now an official CNCF project.

ARMO Platform

ARMO Platform is the enterprise version of [Kubescape](#). It empowers Kubernetes security practitioners to identify and automatically fix misconfigurations and vulnerabilities throughout the entire CI/CD pipeline from code to cluster. It streamlines the automation of Kubernetes compliance requirements, safeguards workloads in runtime, and allows exploration of RBAC in a whole new way.

By combining context from the Kubernetes control plane, runtime insights, and specific workload data, ARMO Platform enables Kubernetes security professionals to eliminate the security noise in their clusters from thousands of alerts to high-priority threat vectors. It empowers them to dare to remediate security issues without breaking running applications. As a result, teams are not only able to save significant resources and maintain application delivery and agility levels, but also bring their Kubernetes security effectiveness to a new level.

Automated
Kubernetes **security**
and **compliance**

Automated Kubernetes
vulnerabilities
assessment

Kubernetes
RBAC made easy

Block Kubernetes
attack paths

ARMO

The Makers of Kubescape 



Sign up for
ARMO Platform



Get involved
on **GitHub**



Follow us
on **X**



Join the discussion
on **Slack**

