
KubeCon
CloudNativeCon
North America 2024

KubeCon CloudNativeCon
North America 2024

Why Perfect Compliance Is the Enemy of Good Kubernetes Security

Michele Chubirka
Cloud Security Advocate, Google
https://linktr.ee/chubirka

Topics

- Security frameworks and standards
- Compliance culture vs security architecture
- Control domains and security principles
- Kubernetes architectural security concepts
- Security as a product feature of the platform

What is compliance?

Compliance vs security architecture

"...consider the brakes on a car...**having better brakes enables the car to be driven at much higher speeds, because the driver now has the confidence that ... braking will be fast and efficient.** It is a completely different way of viewing the same function – one way is about reducing overall speed, the other about increasing it."

Sherwood, J., Clark, A., & Lynas, D. (2005). *Enterprise security architecture : a business-driven approach*. Cmp Books.

Security control domains

Control domains help align Kubernetes security with a security program.

Group hardening/controls according to the domains from a framework.

This action helps you document security controls and any gaps.

Document these in a control mapping matrix for compliance teams.

	A	B	C	D	E	F
1	**PCI DSS Requirements v3.2.1**	**Milestone**	**Status** *Please enter "yes" if fully compliant with the requirement*	**If status is "N/A", please explain why requirement is Not Applicable**	**If status is "No", please complete the**	
2					**Stage of Implementation**	**Estimated Date for Completion of Milestone**
75	***Requirement 6: Develop and maintain secure systems and applications***					
76	**6.1** Establish a process to identify security vulnerabilities, using reputable outside sources for security vulnerability information, and assign a risk ranking (for example, as "high," "medium," or "low") to newly discovered security vulnerabilities. ***Note:*** *Risk rankings should be based on industry best practices as well as consideration of potential impact. For example, criteria for ranking vulnerabilities may include consideration of the CVSS base score, and/or the classification by the vendor, and/or type of systems affected.* *Methods for evaluating vulnerabilities and assigning risk ratings will vary based on an organization's environment and risk-assessment strategy. Risk rankings should, at a minimum, identify all vulnerabilities considered to be a "high risk" to the environment. In addition to the risk ranking, vulnerabilities may be considered "critical" if they pose an imminent threat to the environment, impact critical systems, and/or would result in a potential compromise if not addressed. Examples of critical systems may include security systems, public-facing devices and systems, databases, and other systems that store, process, or transmit cardholder data.*	**3**				
77	**6.2** Ensure that all system components and software are protected from known vulnerabilities by installing applicable vendor-supplied security patches. Install critical security patches within one month of release. Note: Critical security patches should be identified according to the risk ranking process defined in Requirement 6.1.	**3**				
78	**6.3** Develop internal and external software applications (including web-based administrative access to applications) securely, as follows: • In accordance with PCI DSS (for example, secure authentication and logging) • Based on industry standards and/or best practices. • Incorporating information security throughout the software-development life cycle ***Note****: This applies to all software developed internally as well as bespoke or custom software developed by a third party.*	**3**				
79	**6.3.1** Remove development, test and/or custom application accounts, user IDs, and passwords before applications become active or are released to customers.	**3**				
80	**6.3.2** Review custom code prior to release to production or customers in order to identify any potential coding vulnerability (using either manual or automated processes) to include at least the following: • Code changes are reviewed by individuals other than the originating code author, and by individuals knowledgeable about code-review techniques and secure coding practices. • Code reviews ensure code is developed according to secure coding guidelines • Appropriate corrections are implemented prior to release. • Code-review results are reviewed and approved by management prior to release. ***Note****: This requirement for code reviews applies to all custom code (both internal and public-facing), as part of the system development life cycle. Code reviews can be conducted by knowledgeable internal personnel or third parties. Public-facing web applications are also subject to additional controls, to address ongoing threats and vulnerabilities after implementation, as defined at PCI DSS Requirement 6.6.*	**3**				
81	**6.4** Follow change control processes and procedures for all changes to system components. The processes must include the following:	**3**				
82	**6.4.1** Separate development/test environments from production environments, and enforce the separation with access controls.	**3**				
83	**6.4.2** Separation of duties between development/test and production environments	**3**				
84	**6.4.3** Production data (live PANs) are not used for testing or development	**3**				

Control Mapping Example: PCI DSS Prioritized Approach

Google Cloud

Key security principles

Access control - Intersection of identity and data governance

Immutability and ephemerality - Kubernetes workloads are unchanging and short-lived*

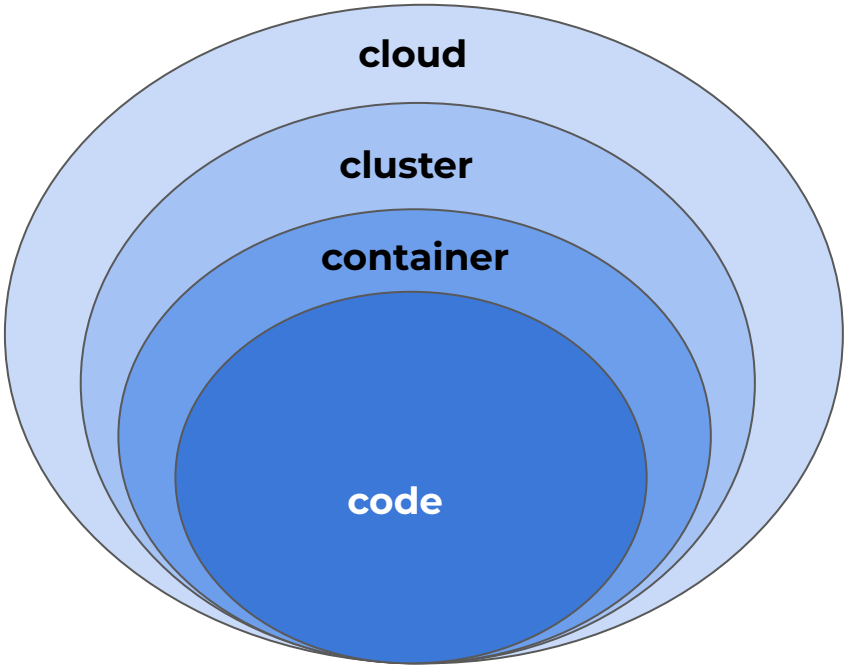
Least-privilege - The least amount of access needed for a principal.

Separation-of-duties - Limit sets of permissions to ensure a single principal cannot abuse a system.

Defense-in-depth - Apply multiple controls to achieve comprehensive security in the event one control fails.

**There are instances where this may not be true, specifically with data science. use-cases.*

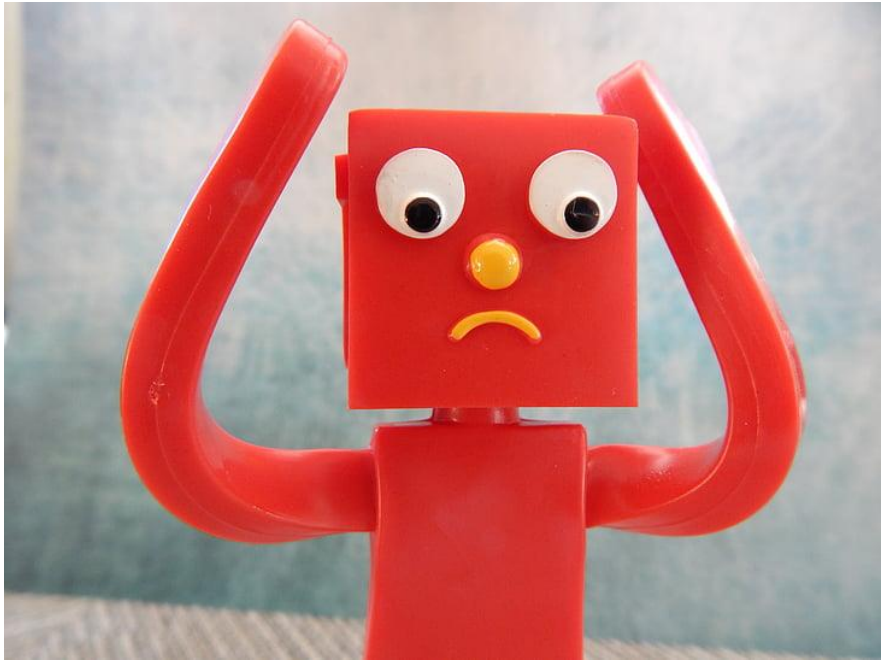
Cloud native security model: 4 Cs



- Consider the 4 Cs of cloud native security when building your Kubernetes threat model.
- What controls will you add to protect each layer?

Kubernetes architectural security concepts

- Identity and credential management
- Multi-tenancy/resource segregation
- Container image assurance
- Secure SDLC
- Runtime security
- Logging and monitoring
- Conway’s security law



Identity and credential management concepts

Two primary identities in Kubernetes

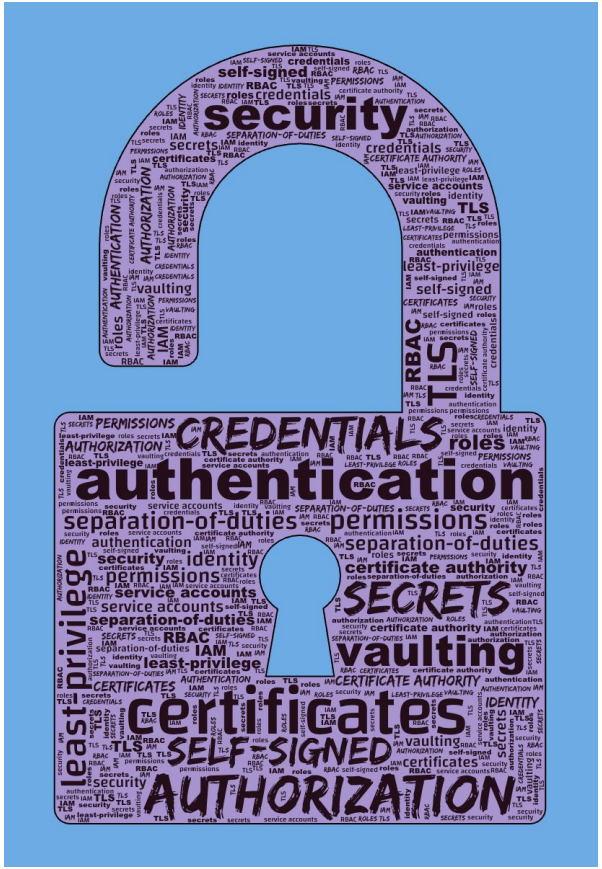
- Humans (admins and developers)
- Non-interactive (service accounts/workloads)

Many items in the benchmark focus on least-privilege and authN/authZ.

Design your identity, certificate and credential management to support security principles.

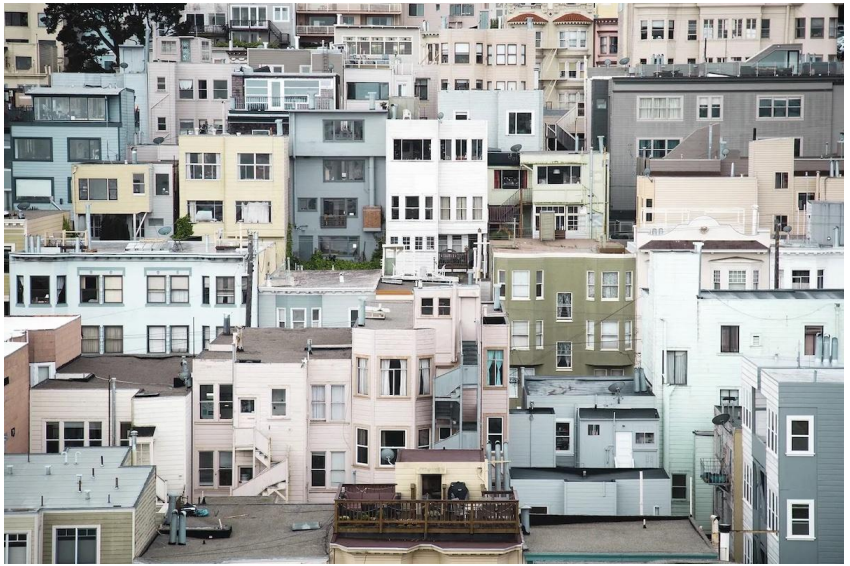
Identity and credential considerations

- How will you implement least-privilege? Who will maintain it?
- RBAC profiles cover interactive and non-interactive identities, don't mix them.
- Kubernetes is a big API dependent on authN ***and*** authZ.
- Certificates are required for TLS and mTLS, how will you provision them?
- Kubernetes secrets vs vaulting



Multi-tenancy and resource segregation principles

- Design clusters/fleet based on trust boundaries and least-privilege.
- Access control is the intersection of identity and data governance.
- Minimize risk of lateral movement and protect individual tenant resources.
- Identify requirements for hard vs soft multi-tenancy



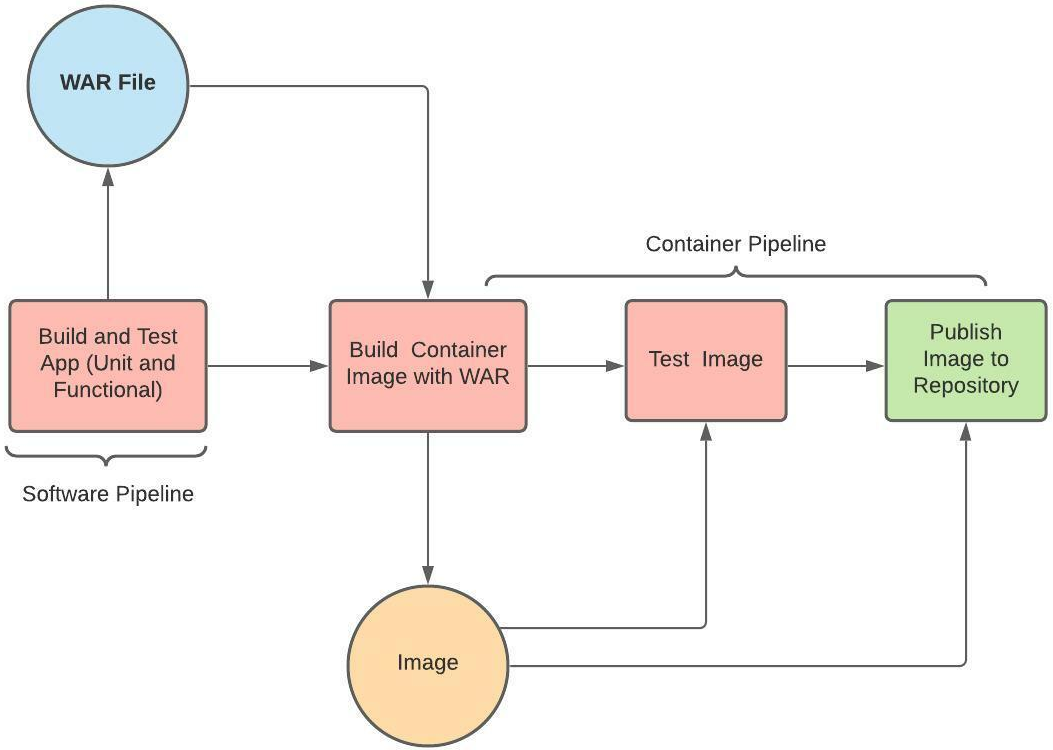
Multi-tenancy and resource segregation considerations

- Protection/isolation of the control plane from the data plane.
- Use separate clusters, node isolation, namespaces, network policies, container sandboxing, MAC, and/or cgroups to create boundaries
- Group pods with similar purposes or access types together.
- Segregate by data governance profile, i.e. PCI vs non PCI, non-prod vs prod, external vs internal access,
- Separate clusters for different business units.
- Keep customer applications isolated from internal applications.

Container image assurance considerations

- Every unnecessary process in a container increases the platform risk.
- The running instance is not the same as the container image.
- A container is just a software package, including its dependencies, that run in a dedicated area of user space.

Container supply chain



Secure container SDLC principles

Trust	Use a trusted source for base images. Create trusted images through signing.
Validate	Validate the image with a security tool to identify vulnerabilities in the libraries or config. Include cryptographic validation.
Economize	Only add elements necessary for your microservice.
Reduce	Follow least privilege: don't run root processes, don't run the container as privileged.
Limit	Limit syscalls and Linux Capabilities.
Parameterize	Don't hardcode configs or embed credentials/secrets in the image, parameterize or vault instead.
Immutable	Don't add a shell or other interactive tools, don't login to a running instance.
Minimize	Use techniques like Distroless, rootless, or "from scratch," to eliminate unnecessary elements, layers, and privileges.
Automate	Use automation to ensure security validation is automatically performed with every build or change to the image.

Runtime security principles

- Kubernetes becomes the hall monitor for containers through admission control.
- Enforce immutability and ephemerality.
- The majority of container and Kubernetes security should be focused on image assurance, which is preventative.

Runtime security concerns

- Choose a container runtime supported by your security tools.
- If using seccomp, MAC (AppArmor/SELinux), and linux capabilities, consider the impact to dev teams.
- Include container drift detection and prevention.
- Implement pre-deploy runtime policies through pod security standards and admission control.
- Identify how you will move from warn/audit to enforcement.

Admission policy criteria

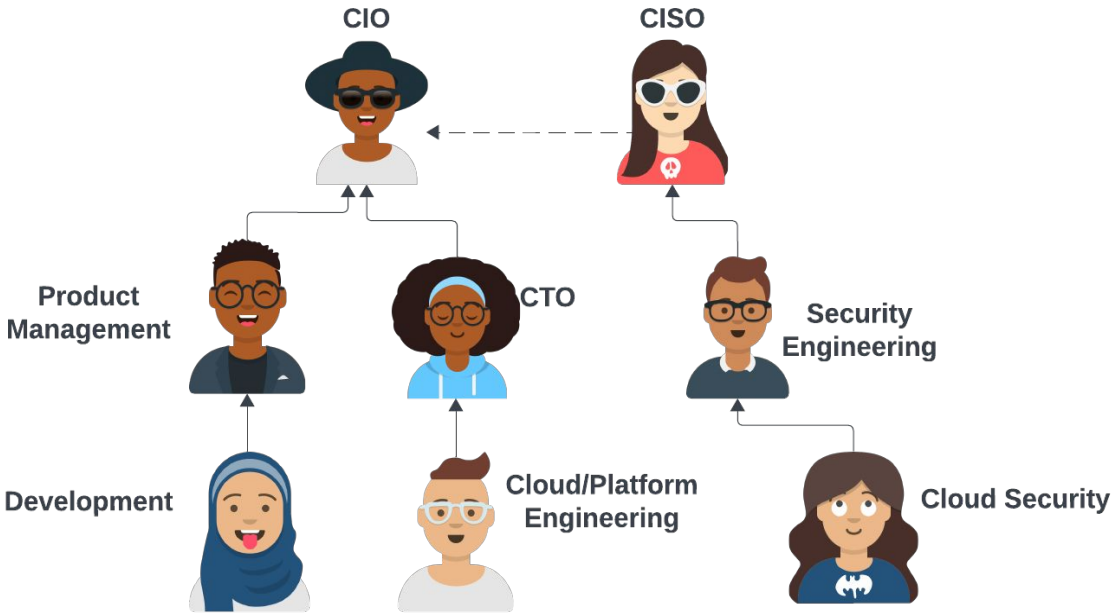
- Is the image known (checksum) and from a trusted repository?
- Is the image approved i.e. has it passed the security policy release criteria?
- Will you break deploys?
- What about image TTLs and expiration of running instances?
- When a security capability enforces a runtime control, who will you alert?

Monitoring and logging architecture considerations

- Two main types of logs:
  - system component and platform
  - application
- Where do you log, how do you log, what do you log and how much (i.e. what level)?
- Security audit records track users, processes, and applications that touch the Kubernetes API.
- How will all this data be correlated?
- Will you include resource/full metrics?

Pragmatism and Conway's security law

Your Kubernetes security will mirror your org chart.



Kubernetes security as a platform feature

- Integrate security principles and capabilities as features of the platform.
- Implement “compliance as property,” making it everyone’s responsibility.
- Negotiate compliance requirements, where possible, with security teams.
- Choose simple, manageable solutions over complex ones.
- Use a RACI matrix to establish who is responsible for security tasks.
- Kubernetes security is an adaptive challenge requiring conflict management skills.

References

Kubernetes Security Documentation https://kubernetes.io/docs/concepts/security/
Kubernetes RBAC best practices
https://kubernetes.io/docs/concepts/security/rbac-good-practices/
Kubernetes Multi-Tenancy https://kubernetes.io/docs/concepts/security/multi-tenancy/
Cloud Native Security and Kubernetes
https://kubernetes.io/docs/concepts/security/cloud-native-security/
CNCF Security Whitepaper
https://www.cncf.io/wp-content/uploads/2022/06/CNCF_cloud-native-security-whitepaper-May2022-v2.pdf
A Deep Dive into Kubernetes Threat Modeling
https://www.trendmicro.com/vinfo/us/security/news/security-technology/a-deep-dive-into-kubernetes-threat-modeling
CIS Kubernetes Benchmark https://www.cisecurity.org/benchmark/kubernetes
NSA Kubernetes Hardening Guide
https://media.defense.gov/2022/Aug/29/2003066362/-1/-1/0/CTR_KUBERNETES_HARDENING_GUIDANCE_1.2_20220829.PDF


KubeCon | CloudNativeCon
North America 2024

Thank you!

Michele Chubirka
Cloud Security Advocate, Google
https://linktr.ee/chubirka