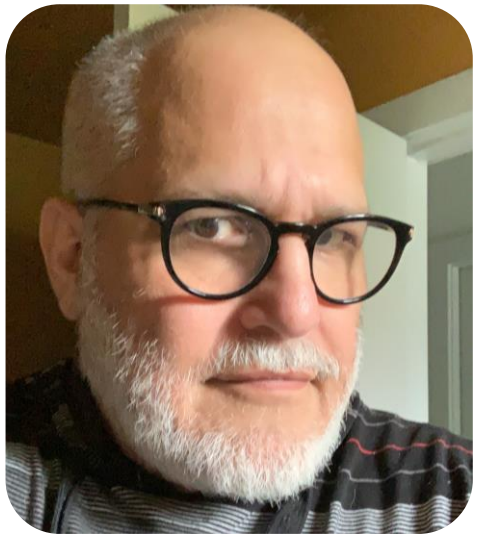
October 21, 2020

# The abc’s of Kubernetes Security

CNCF Webinar

# Contact



Roger Klorese

Senior Product Manager

roger.klorese@suse.com

Danny Sauer

Senior Software Engineer

dsauer@suse.com





SUSE

# Agenda

1 How secure do we feel about containers?

2 Addressing container security

3 Governance

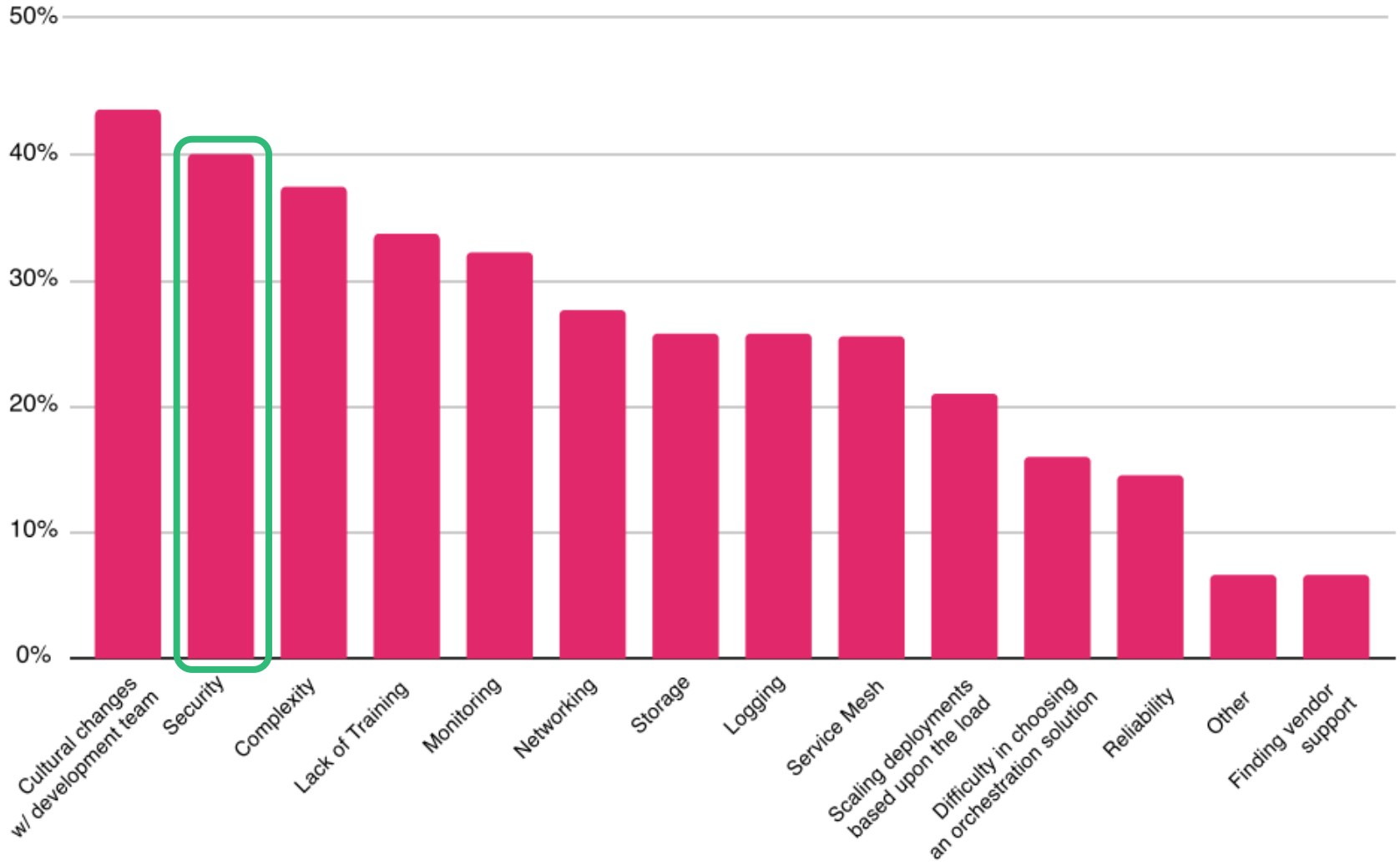
4 Where do I start?

# How secure do we feel about containers?

# True or false?

Containers are inherently insecure.

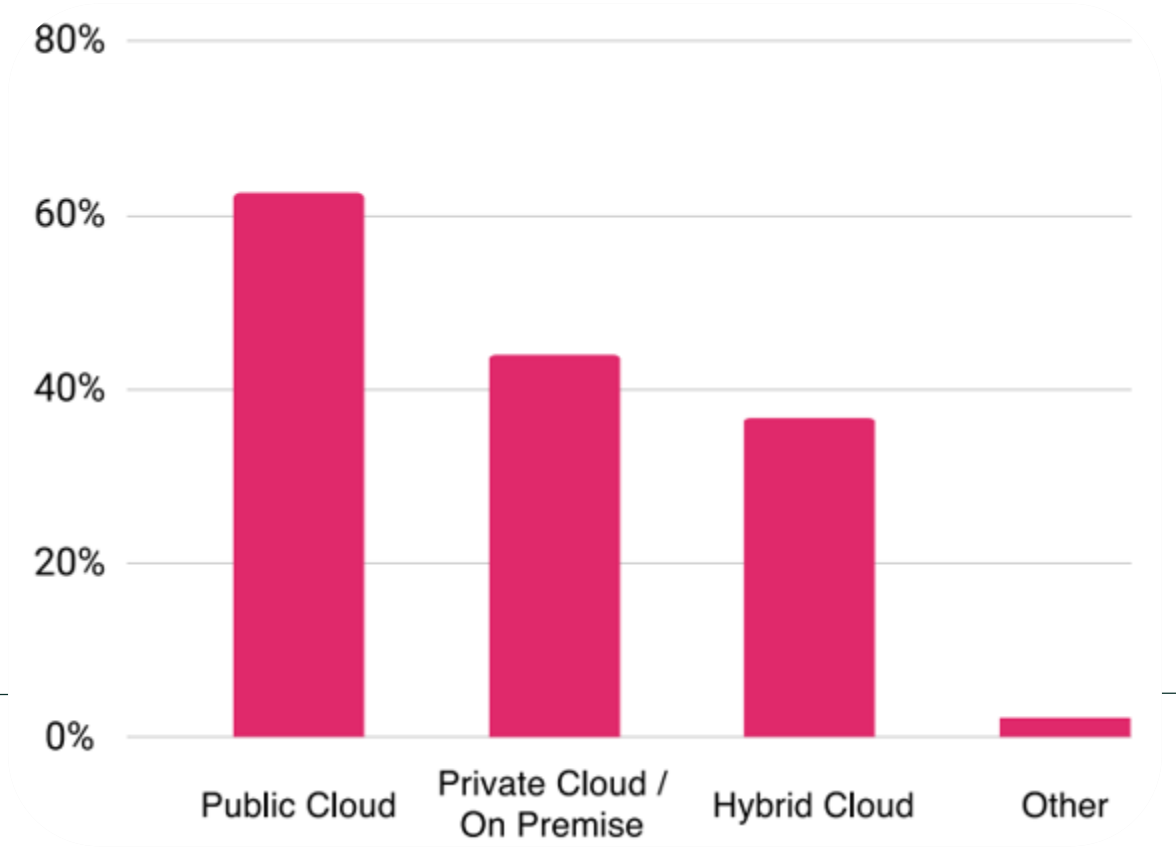
# What are your challenges in using / deploying containers?




40% Security

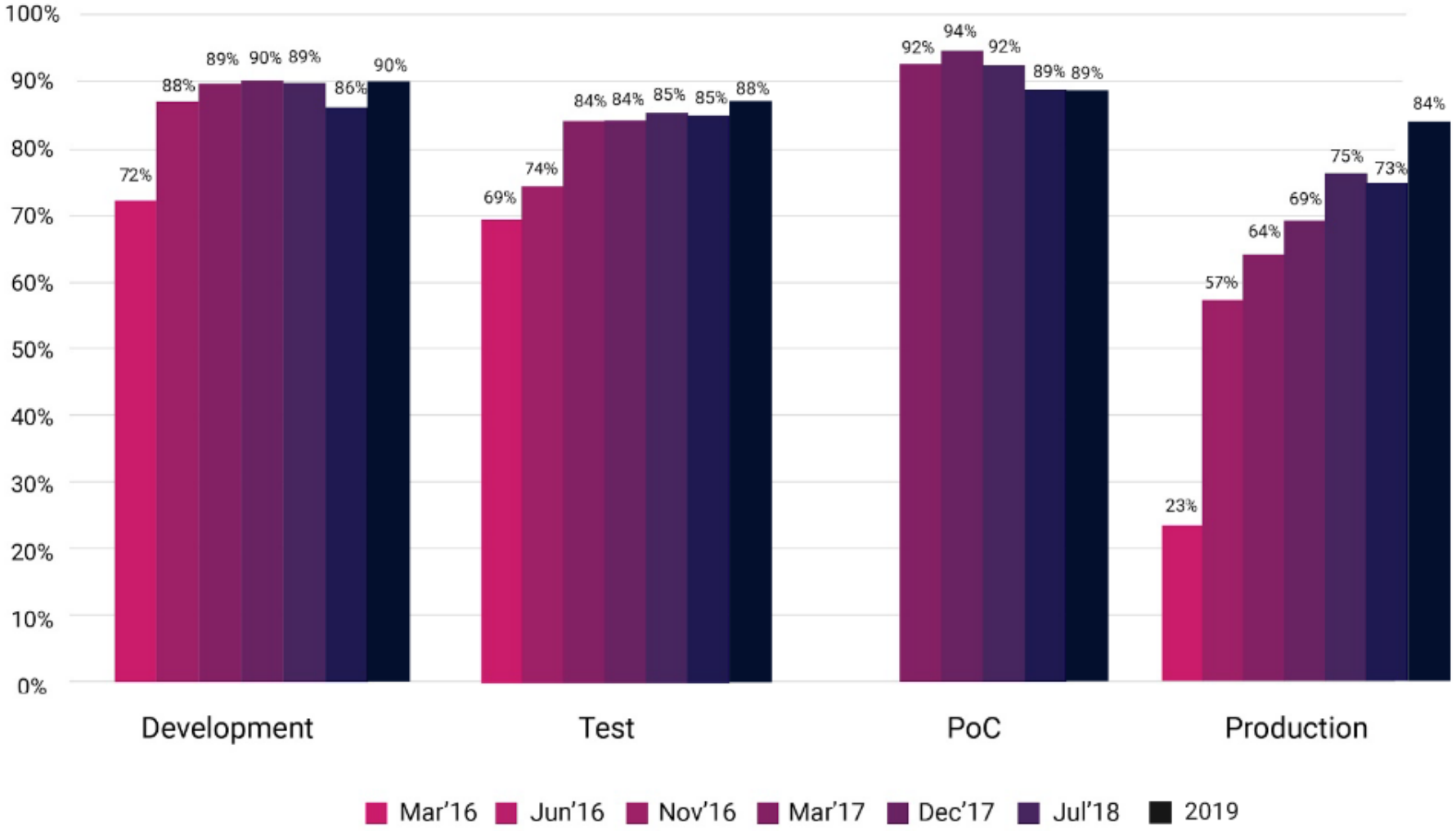
CNCF user Survey 2019
https://www.cncf.io/wp-content/uploads/2020/03/CNCF_Survey_Report.pdf

# Where are you deploying most?




CNCF user Survey 2019
https://www.cncf.io/wp-content/uploads/2020/03/CNCF_Survey_Report.pdf

# Container use since 2016




CNCF user Survey 2019
https://www.cncf.io/wp-content/uploads/2020/03/CNCF_Survey_Report.pdf

# Some learnings from an enterprise study

94%: "Containers have security implications"

31%: "Worried about the lack of mature security solutions for containers"

31%: "Current server security solutions do not support containers"

28%: "A single infected container could easily spread to others"

16%: "Portability of containers means they could be more susceptible to 'in motion' compromise"

ESG Strategy Group - Threat Stack Cloud Security Report 2017: Security at Speed & Scale

# Addressing container security

# Security requirements

- Enforcing the deployment of a secure gold image on container hosts, using governance and policies.
- Role-based access control to the platform itself and the containers.
- Runtime and at-rest scanning.
- Network segmentation and access control.
- Network visibility.
- Encryption of data in motion within and between clusters.
- Secrets management, to avoid having secrets such as database passwords in container images.
- Policies and underlying operating system tools to restrict runtime access.
- Monitoring the security posture of the platform, using classical security tools.

# Secure gold image

*Enforcing the deployment of a secure gold image on container hosts, using governance and policies*

Best Practice:

- Build gold master container image based on tested OS base containers
- Integrate CI/CD pipeline to deliver applications and app updates consistently and securely

# Role-based access control

*Role-based access control to the platform itself and the containers*

Best Practice:

(In decreasing order of security)

- Create service account for application with only the permissions it needs
- Grant admin access to the default service account for a particular namespace to that same application namespace
- Create service account for application that has admin access to the application's namespace

WORST Practice:

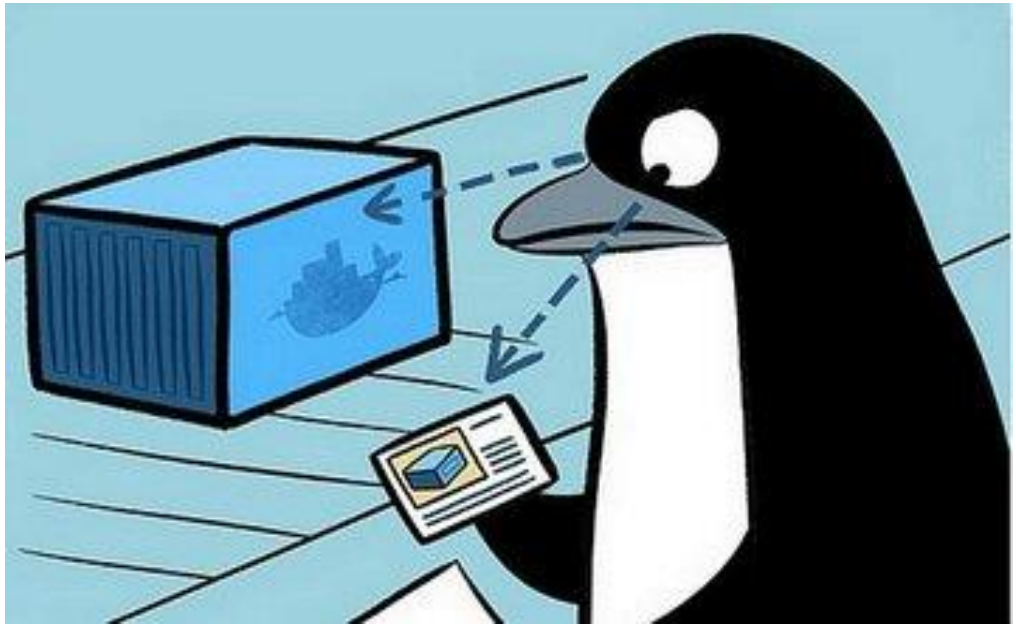
- Disable RBAC, or grant all permissions on workloads to kube-system

# Scanning

*Runtime and at-rest scanning.*

Best Practice:

- Build containers with methodology that performs at-rest scanning
- Scanners integrated into CI/CD pipeline
- Some scanners also perform runtime scanning



# Network policies

*Network segmentation and access control.*

Best Practice:

- Use a CNI plugin that implements network policies to:
  - Control ingress and egress to the clusters
  - Control ingress and egress to namespaces
- If your distribution does not support network policies:
  - Consider products such as container firewalls

# Observability

*Network visibility. (and more)*

Best Practice:

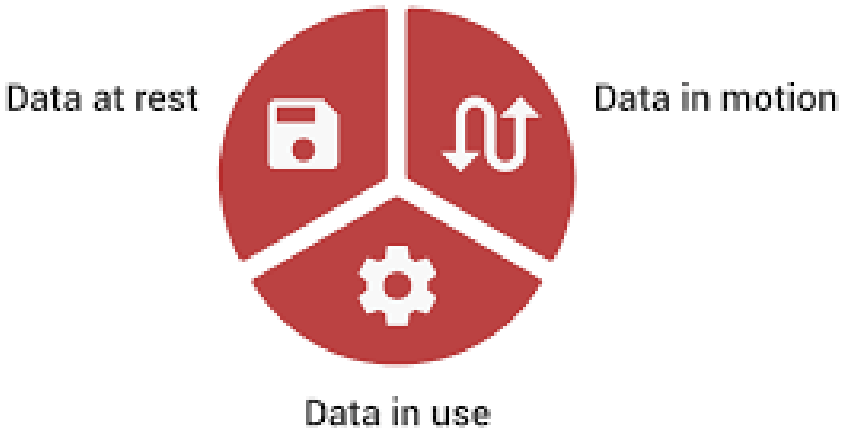
- Monitor network traffic, security, and performance:
  - Prometheus with appropriate exporters
  - CNI-specific network flow analysis
  - Service mesh

# Encryption in motion

*Encryption of data in motion within and between clusters.*

Best Practice:

- Utilize the in-motion encryption within the cluster delivered by default with cluster-signed certificates
- Add customer-supplied trusted-root certificates for external interfaces (API-server, directory services, etc.)
    - If possible, integrate auto-issuance and auto-renewal



# Secrets management

*Secrets management, to avoid having secrets such as database passwords in container images.*

Best Practice:

- Keep secrets in memory for as little time as possible
  - Use secrets managers (e.g., HashiCorp Vault)
  - Access secrets from environment variables
- If you must use mounted secrets, enable encryption at rest

# Runtime security

*Policies and underlying operating system tools to restrict runtime access.*

Best Practice:

- Use Pod Security Policies (PSPs) to control:
  - Use of privileged containers
  - Use of host resources (file systems, networks, etc.)
  - Privilege escalation
  - Linux capabilities
  - OS security profiles
- Consider use of runtime security monitoring tools

# Platform security

*Monitoring the security posture of the platform, using classical security tools.*

*Don't forget there is a platform underneath the container environment!*

Best Practice:

- OS-level security tools and profiles
- Physical and virtual network security tools:
  - Firewalls, WAF, IDS/IPS, anti-malware
- Storage and cloud security policies

# Host Hardening and Security Practices

Signed and Secure software packages from SUSE Build Services

UEFI Secure Boot

OpenSCAP – Security Standards Automation

Firewall

Transport Layer Security

SSSD for identity and authentication resources (Active Directory)

Linux Audit Subsystem

Process Hardening

AppArmor / SELinux

Filesystem Encryption

Live Kernel Patching for reduced downtime

Security Updates and backporting to previous versions

# Governance

# Governance examples

- Containers cannot be started by a user using a shell on the host or by remote CLI.
- A set of workloads should run on the same hosts (affinity) or cannot run on the same host (anti-affinity).
- Kubernetes deployment can only be created using Helm.
- Transmission between nodes should be encrypted.
- Data at rest should be encrypted.
- Secrets should be centrally managed and encrypted.
- Only specific groups of users can start and stop containers belonging to a particular application (RBAC applied to scheduling).
- Certain apps need a dedicated namespace.
- YAML files must be managed subject to revision control and RBAC.

# Where do I start?

# The "low-hanging fruit"

- Disable anonymous access
- Disable automounting `default` service account token
- Use admission control to block privilege escalation by shell access on privileged containers
- Limit user impersonation
- Disallow privileged containers – or if needed, control individual privileges
- Disallow or restrict sharing of host PID namespace, IPC namespace, and network stack
- Use resource limits to mitigate "noisy neighbor syndrome"
- Patch promptly!
- TRAIN DEVS AND DEVOPS IN SECURITY CONSIDERATIONS! (DevSecOps at the heart of the process)

# Staying vigilant

- Don't release, reconfigure, etc. without security analysis
- `kube-bench*` early and often!
- Integrate automated testing and continuous assessment
  - Don't rely on the most vulnerable component – humans!


* An example of a security analysis tool - there are others. But Aqua's implementation of The Center for Internet Security (CISecurity) benchmarks is a great choice.

For more information, contact SUSE at:

+1 800 796 3700 (U.S./Canada)

+49 (0)911-740 53-0 (Worldwide)

SUSE

Maxfeldstrasse

90409 Nuremberg

www.suse.com