



# Container & Kubernetes Security for Developers

Andreas Falk

**DEVOXX™**  
**UKRAINE**

# About Me

---

Andreas Falk  
Novatec Consulting  
(Stuttgart, Germany)

 [andreas.falk@novatec-gmbh.de](mailto:andreas.falk@novatec-gmbh.de)  
 [@andifalk](https://twitter.com/andifalk)



# Security @ Novatec

---

## Our Offerings:

- OAuth 2.0 & OpenID Connect
  - Trainings, Audits and Coaching
  - Support in Architecture and Implementation
- Threat Modeling
  - Trainings & Workshops



<https://www.novatec-gmbh.de/en>

<https://www.novatec-gmbh.de/en/consulting/oauth-2-0-and-openid-connect/>

# Agenda

---

1. Use Cases for Containers
2. What Can Go Wrong
3. The Path for Secure Development
  - Application Security
  - Container Security
  - Kubernetes Security
  - Kubernetes Secrets



# Slides and the Code?

---

Look here:

## Demos

### Iteration 1: Application Security

- Hello Spring Boot

### Iteration 2: Container Security

- Root Container
- Rootless Container
- Rootless Container with JIB

### Iteration 3: Kubernetes Security

- Initial Unsafe Kubernetes Deployment
- Safe Kubernetes Deployment (Pod Security Context)
- Safe Kubernetes Deployment (Pod Security Policy)
- Safe Kubernetes Deployment (Open Policy Agent)

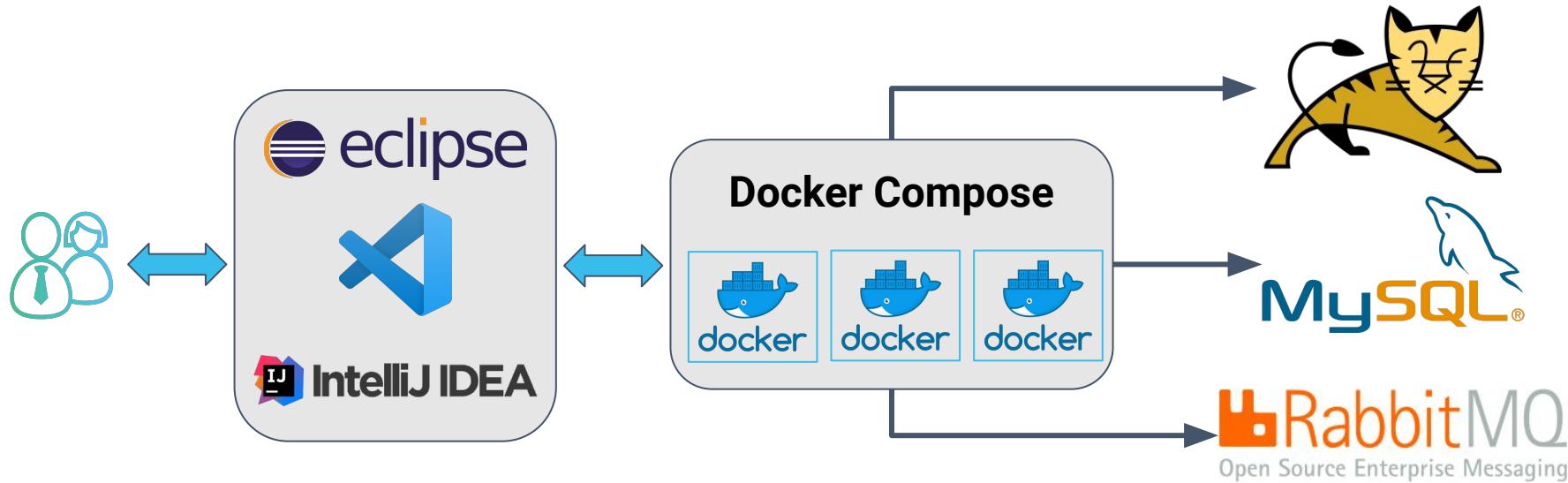
<https://github.com/andifalk/secure-development-on-kubernetes>

---

# Container Use Cases

# Development Environments

---



# Integration Testing (TestContainers)

---



- Java library that supports JUnit tests
- Provides lightweight, throwaway instances of common 3rd party components that can run in a Docker container.
- Databases (DB2, Oracle XE, MS SQL, MySQL, ...)
- Kafka, RabbitMQ
- Hashicorp Vault, Keycloak
- ...

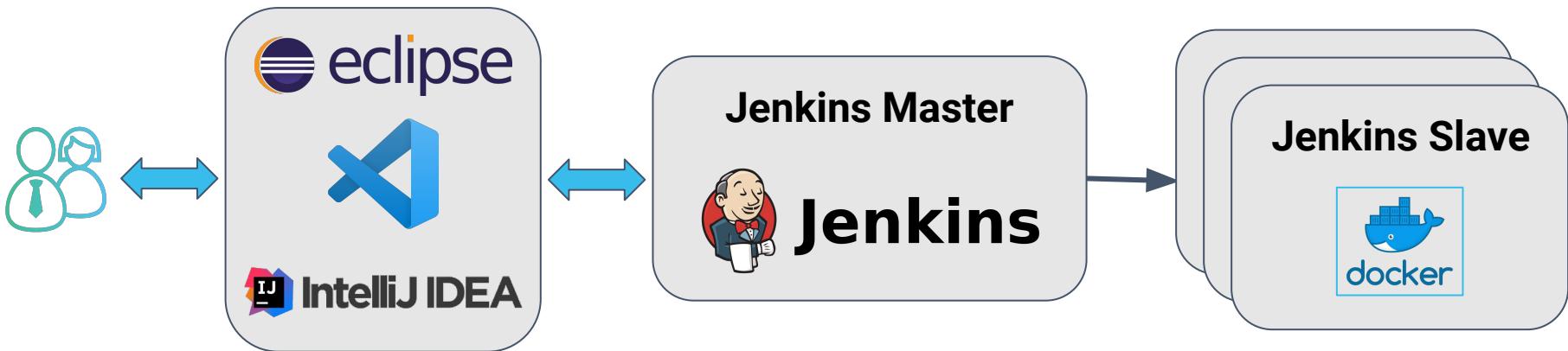
<https://www.testcontainers.org>

# CI-/CD Pipelines

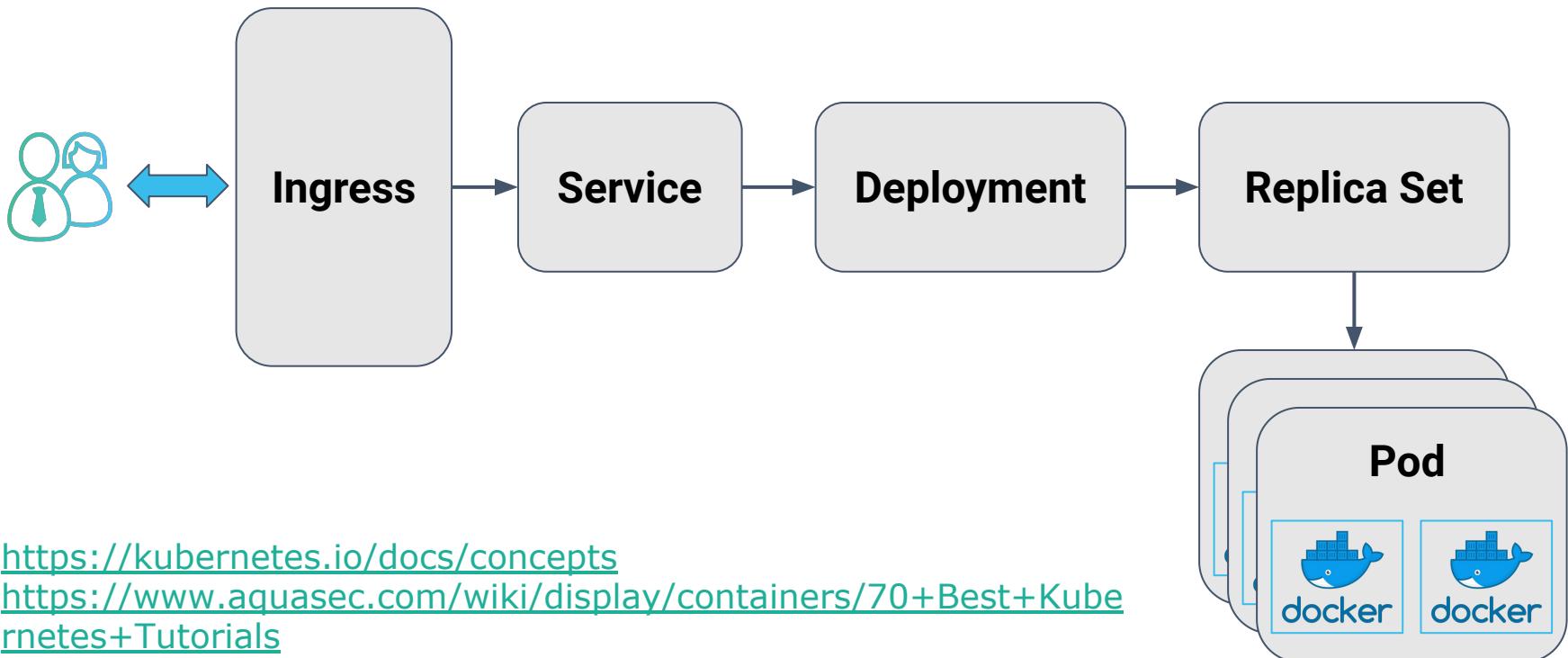
---



JENKINS X



# Container & Kubernetes (Production)



<https://kubernetes.io/docs/concepts>

[https://www.aquasec.com/wiki/display/containers/70+Best+Kube  
rnetes+Tutorials](https://www.aquasec.com/wiki/display/containers/70+Best+Kubernetes+Tutorials)

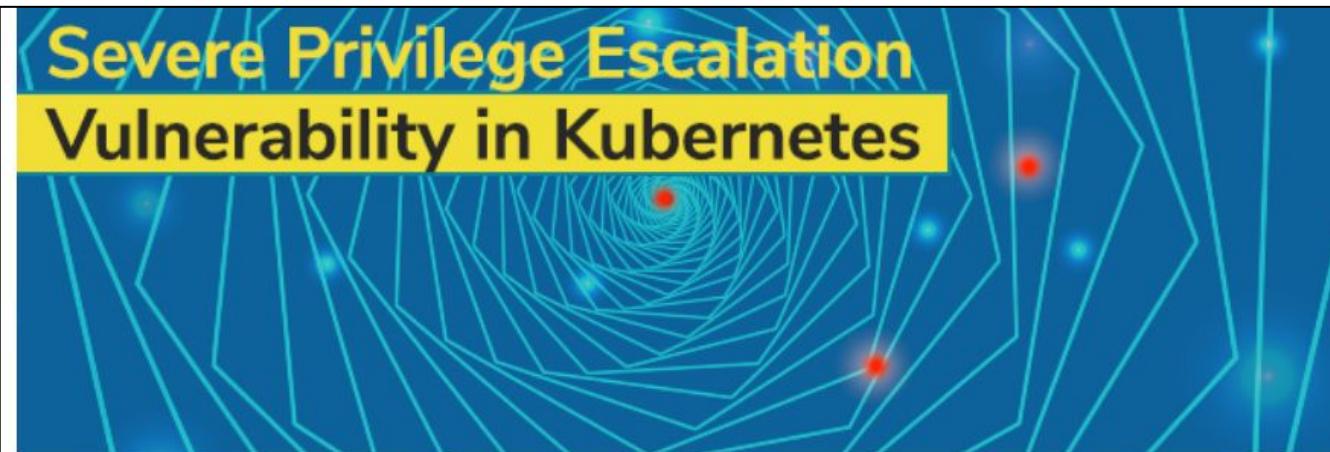
---

# What can go wrong?

Introduction

# Severe Vulnerability in Kubernetes

Source: <https://blog.aquasec.com>



Ariel Shuper • December 06, 2018

## Severe Privilege Escalation Vulnerability in Kubernetes (CVE-2018-1002105)

Earlier this week, a [severe vulnerability in Kubernetes](#) (CVE-2018-1002105) was disclosed that allows an unauthenticated user to perform privilege escalation and gain full admin privileges on a cluster. The CVE was given the high severity score of 9.8 (out of 10) and it affects all Kubernetes versions from 1.0 onwards, but fixes are available for recent versions.

# Crypto Mining Via K8s Dashboard

Source: <https://blog.heptio.com>

## On Securing the Kubernetes Dashboard



Joe Beda [Follow](#)

Feb 28, 2018 · 13 min read

Recently Tesla (the car company) was alerted, by security firm RedLock, that their Kubernetes infrastructure was compromised. The attackers were using Tesla's infrastructure resources to mine cryptocurrency. This type of attack has been called "cryptojacking".

The vector of attack in this case was a Kubernetes Dashboard that was exposed to the general internet with no authentication and elevated privileges. Not only this, but core AWS API keys and secrets were visible. How do you prevent this from happening to you?

# Open ETCD Ports in Kubernetes (1)

SHODAN etcd port:"2379"  Explore Downloads Reports

Exploits Maps Share Search Download Results Create Report

TOTAL RESULTS 2,450

New Service: Keep track of what you have connected to the Internet. Check

TOP COUNTRIES



Country	Count
China	1,116
United States	541
Germany	138
France	117
Singapore	70

TOP ORGANIZATIONS

Organization	Count
Hangzhou Alibaba Advertisin...	417
Amazon.com	273
Tencent cloud computing	172
China Unicom Beijing	111
Hetzner Online GmbH	54

47.52.241.38

Alibaba  
Added on 2019-07-02 11:19:29 GMT  
 Hong Kong

cloud

etcd  
Name: etcd-hk  
Version: 3.2.6  
Uptime: 47h12m20.876361718s  
Peers: http://10.70.10.205:2380

34.77.57.47

47.57.77.34.bc.googleusercontent.com  
Halliburton Company  
Added on 2019-07-02 11:05:41 GMT  
 United States

etcd  
Name: m3db\_local  
Version: 3.2.10  
Uptime: 118h39m34.598205154s  
Peers: http://0.0.0.0:2380

13.229.135.103

ec2-13-229-135-103.ap-southeast-1.compute.amazonaws.com  
Amazon Data Services Singapore  
Added on 2019-07-02 11:07:34 GMT  
 Singapore, Singapore

etcd  
Name: node1  
Version: 3.1.0  
Uptime: 20m8.52416951s  
Peers: http://node1:2380

<https://shodan.io>

# Open ETCD Ports in Kubernetes (2)



```
$ etcdctl --endpoints=http://xx.xx.xx.xx:2379  
cluster-health
```

member b97ee4034db41d17 is healthy: got healthy  
result  
from http://xx.xx.xx.xx:2379  
cluster is healthy

<https://github.com/etcd-io/etcd/releases>

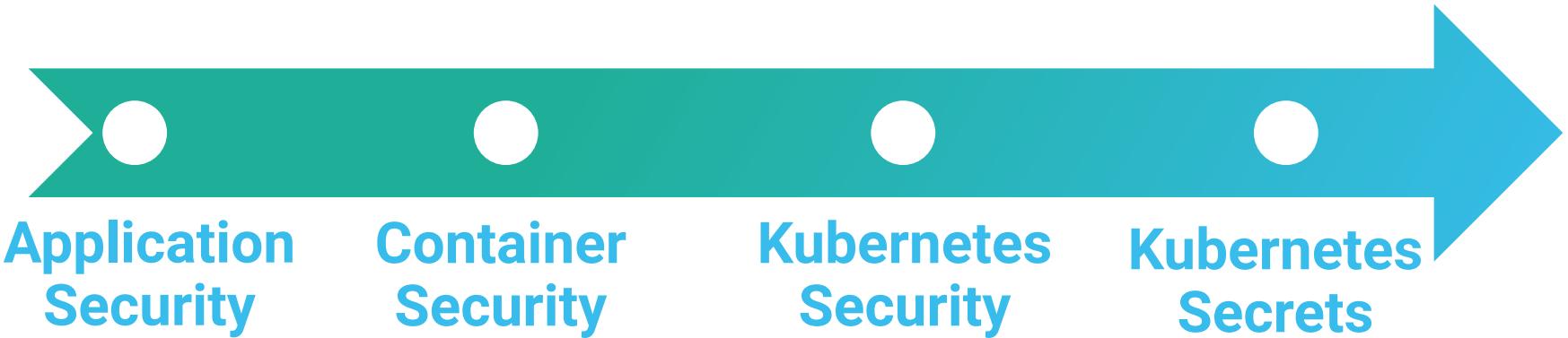
---

# **So what can WE do as Developers?**

**Application- / Docker- / K8s-Security**

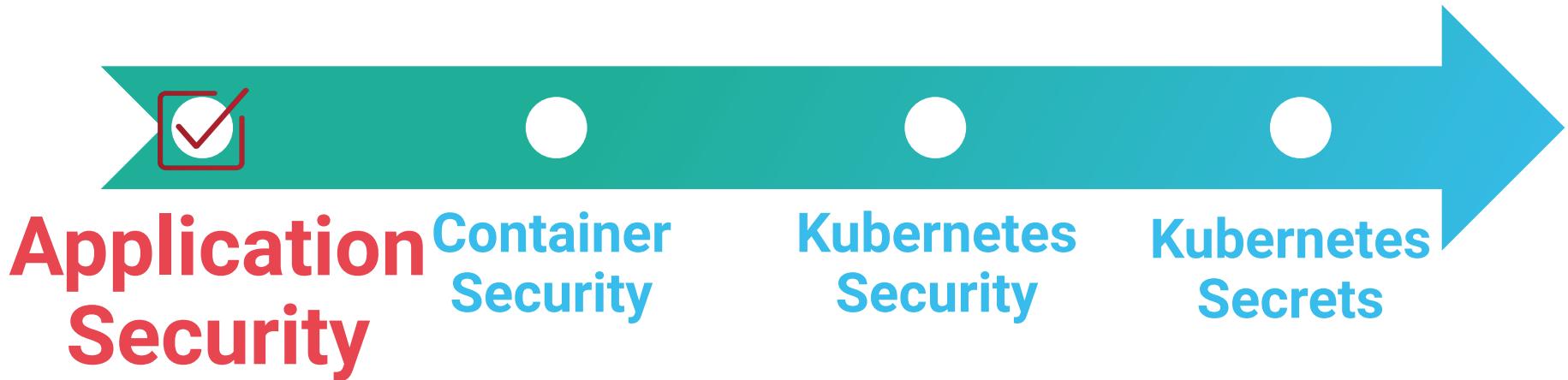
# The Path for Secure Development on K8s

---



# The Path for Secure Development on K8s

---



# Application Security



Authentication



Authorization



SQL Injection



Cross Site Scripting (XSS)



Cross Site Request Forgery (CSRF)



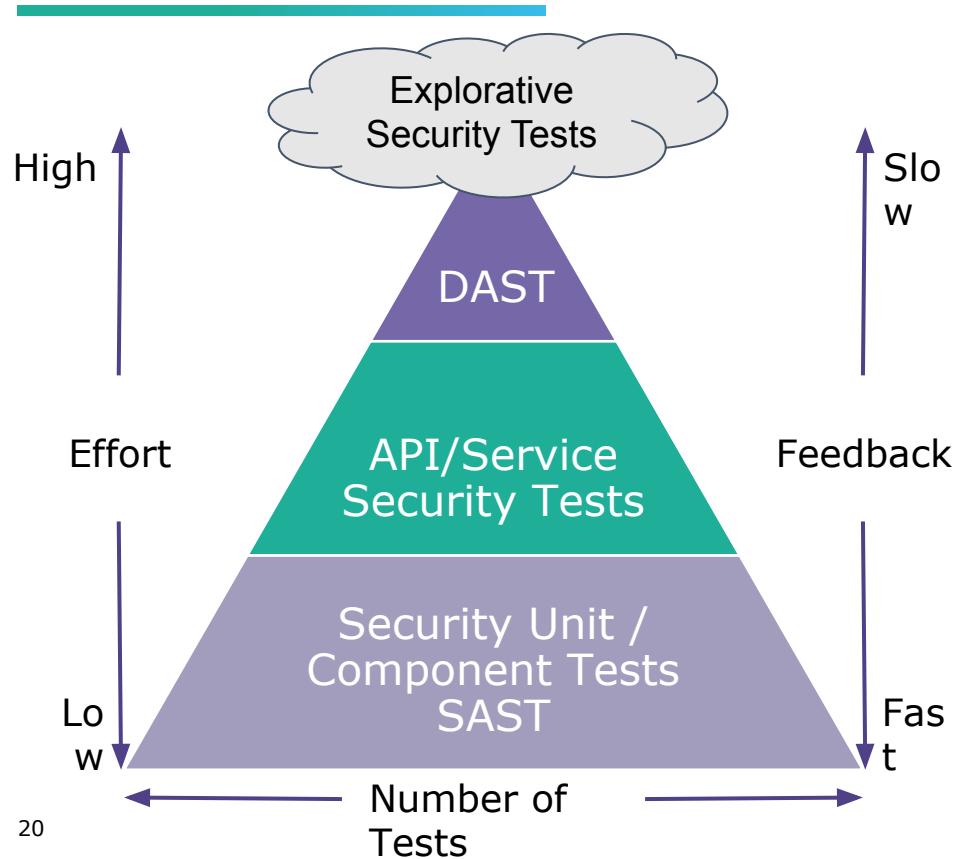
Data Protection (Crypto)



...

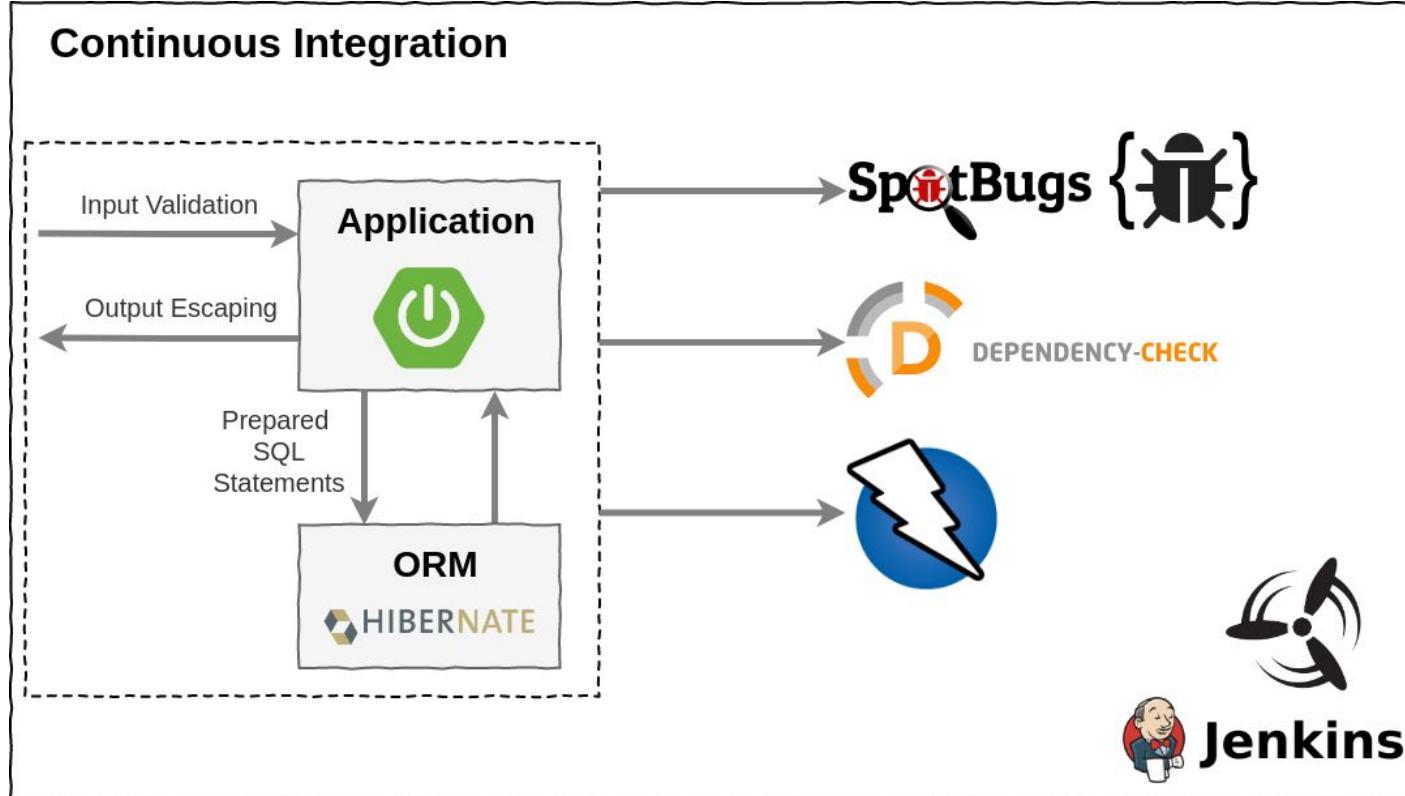


# The Security-Testing Pyramid



- Security Charters in Explorative Testing
- Dynamic Application Security Testing (DAST), e.g OWASP Zap
- Input Validation, Authentication, Authorization, Session Management, Output Escaping (XSS), SQL Injection, Security Response Headers
- Input Validation, Authentication, Output Escaping (XSS), Bypass Business Logic, Error Handling & Logging
- Static Application Security Testing (SAST)

# Automate All The Things!



---

# Live Demo: Show me the code

## Iteration 1: Application Security

<https://github.com/andifalk/secure-development-on-kubernetes>

# Application Security 101

---

- Input Validation for ALL types of input
- Output Encoding to prevent XSS
- Use only Parameterized Queries/Prepared Statements
- Enforce Authentication & Authorization
- Never implement your own Crypto or Session-Management
- Check 3rd Party Dependencies for Vulnerabilities
- Use Static & Dynamic Application Security Testing

<https://cheatsheetseries.owasp.org>

<https://owasp.org/www-project-top-ten>

<https://owasp.org/www-project-proactive-controls>

# The Path for Secure Development on K8s

---



# OWASP Docker Top 10

1. Secure User Mapping
2. Patch Management Strategy
3. Network Segmentation and Firewalling
4. Secure Defaults and Hardening
5. Maintain Security Contexts
6. Protect Secrets
7. Resource Protection
8. Container Image Integrity and Origin
9. Follow Immutable Paradigm
10. Logging

<https://github.com/OWASP/Docker-Security>

<https://doi.org/10.6028/NIST.SP.800-190>

<https://github.com/OWASP/Container-Security-Verification-Standard>

<https://www.bsi.bund.de>

NIST Special Publication 800-190

## Application Container Security Guide

OWASP  
Container Security Verification Standard



Bundesamt  
für Sicherheit in der  
Informationstechnik

SYS: IT-Systeme

SYS.1.6: Container



# Virtual Machine (VM) Basics

vmware®  
Workstation



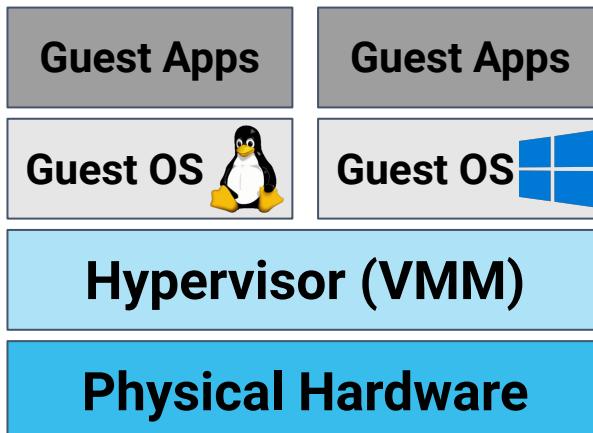
KVM

vmware® ESXi

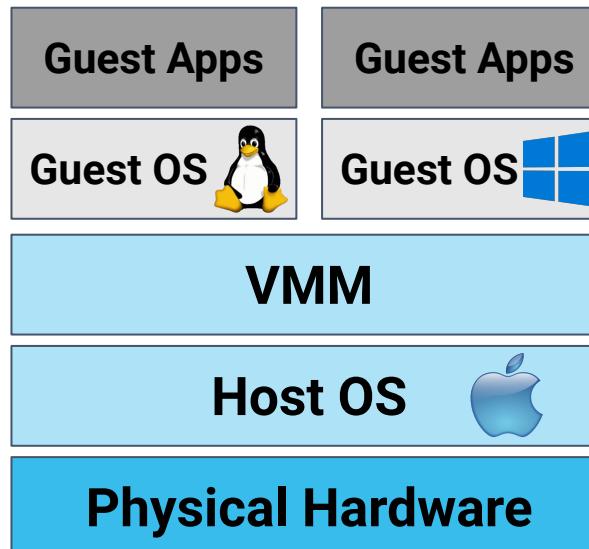
Firecracker



Hyper-V

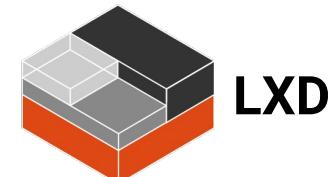
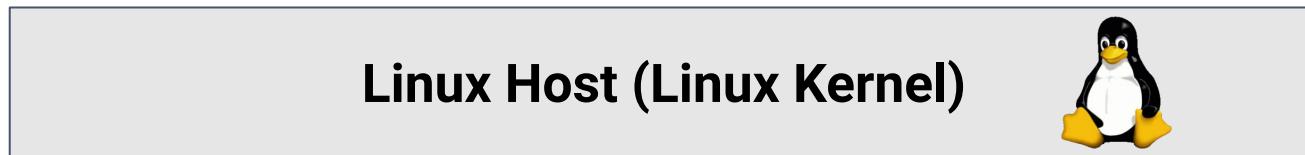
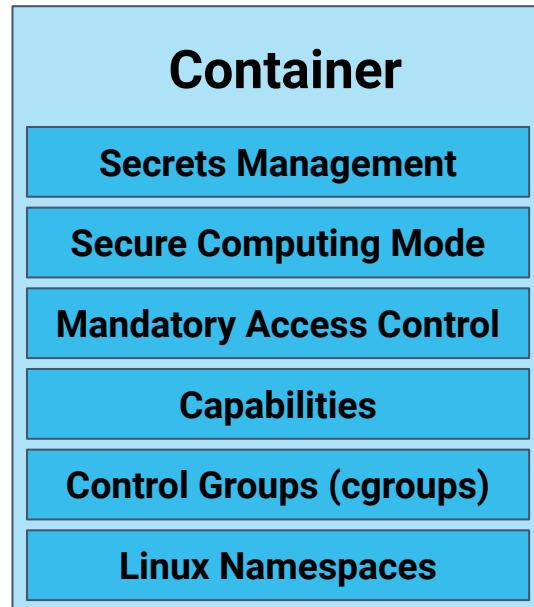
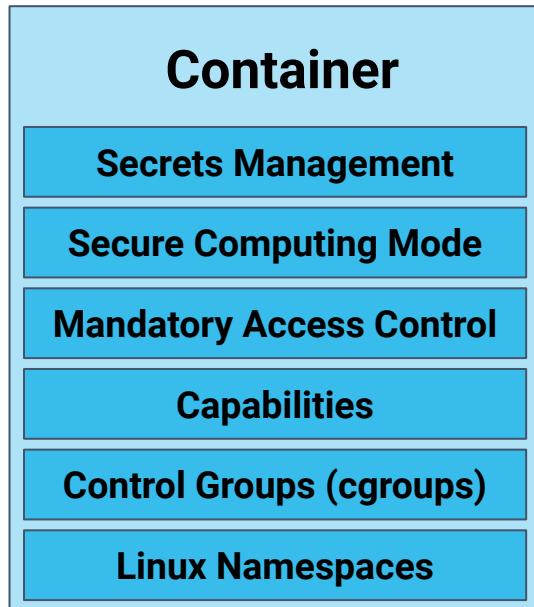


Type 1 Virtual Machine Monitor



Type 2 Virtual Machine Monitor

# Container (Security) Basics



# Linux Kernel Namespaces

---

- Process IDs
- Network
- Mount Points
- Inter-Process Communications (IPC)
- User & Group IDs
- Unix Timesharing System (UTS): hostname & domain names
- Control groups (cgroups)

```
$ man namespaces  
$ sudo lsns
```

# Linux Control Groups (cgroups)

---

- Resource Limits
  - CPU
  - Memory
  - Devices
  - Processes
  - Network

For Java this only works with container aware JDK versions as of **OpenJDK 8u192** or above

**Recommendation:** Use Java 11

# Linux Capabilities

---

- Break up privileges into smaller units
  - CAP\_SYS\_ADMIN
  - CAP\_NET\_ADMIN
  - CAP\_NET\_BIND\_SERVICE
  - CAP\_CHOWN
  - ...

```
$ man capabilities  
$ docker run --cap-drop=ALL --cap-add=NET_BIND_SERVICE
```

<http://man7.org/linux/man-pages/man7/capabilities.7.html>

# Linux Mandatory Access Control & System Calls

---

- Restrict System Calls
  - Secure Computation Mode (seccomp)
  - Google gVisor
- Linux Kernel Security Modules (MAC)
  - AppArmor
  - Security-Enhanced Linux (SELinux)

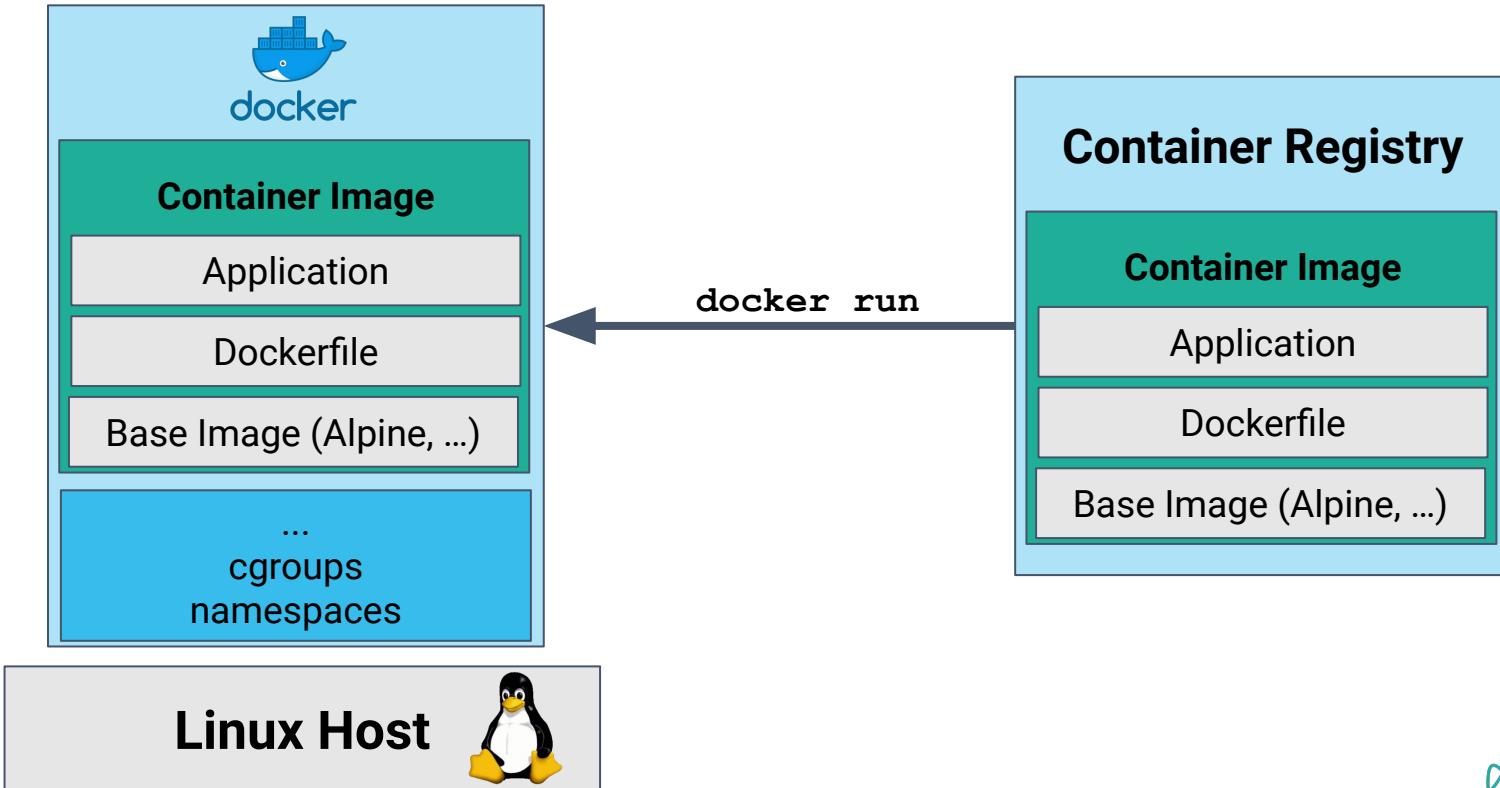
<https://docs.docker.com/engine/security/seccomp>

<https://apparmor.net>

[https://en.wikipedia.org/wiki/Security-Enhanced\\_Linux](https://en.wikipedia.org/wiki/Security-Enhanced_Linux)

<https://gvisor.dev/docs>

# Docker Images



# Breaking Container Isolation

---

- Containers Run as Root by Default
- The --privileged Flag and Capabilities
- Mounting Sensitive Directories
- Mounting the Docker Socket
- Shared Namespaces

<https://docs.docker.com/engine/security/rootless>

<https://docs.docker.com/engine/security/https>

# Mount **sensitive** Directories

---

```
$ touch /ROOT_FOR_HOST  
$ docker run -it -v /:/hostroot ubuntu bash  
root@91083a4eca7d:/$ ls /hostroot
```



```
ROOT_FOR_HOST bin etc home lib  
lib64 media mnt root run srv sys var
```

```
..  
.
```

# All is Root



**CZnative @ home**  
@pczarkowski

Welcome to Kubernetes where everything runs as root and the security doesn't matter!

14:22 - 8. Mai 2019

# Say No To Root (1)

---

## USER directive in Dockerfile

```
FROM openjdk:11-jre-slim
COPY hello-spring-kubernetes-1.0.0-SNAPSHOT.jar app.jar
EXPOSE 8080
RUN addgroup --system --gid 1002 app && adduser
    --system --uid 1002 --gid 1002 appuser
USER 1002
ENTRYPOINT java -jar /app.jar
```

<https://opensource.com/article/18/3/just-say-no-root-containers>

## Say No To Root (2)

---

### Use JIB and Distroless Images

```
plugins {  
    id 'com.google.cloud.tools.jib' version '...' }  
  
jib {  
    container {  
        user = 1002  
    }  
}
```

<https://github.com/GoogleContainerTools/jib>

# Secure Docker Run

---



```
docker run -t -i --privileged ubuntu bash
```



```
docker run -t -i ubuntu bash
```



```
docker run -t -i --security-opt=no-new-privileges ubuntu bash
```



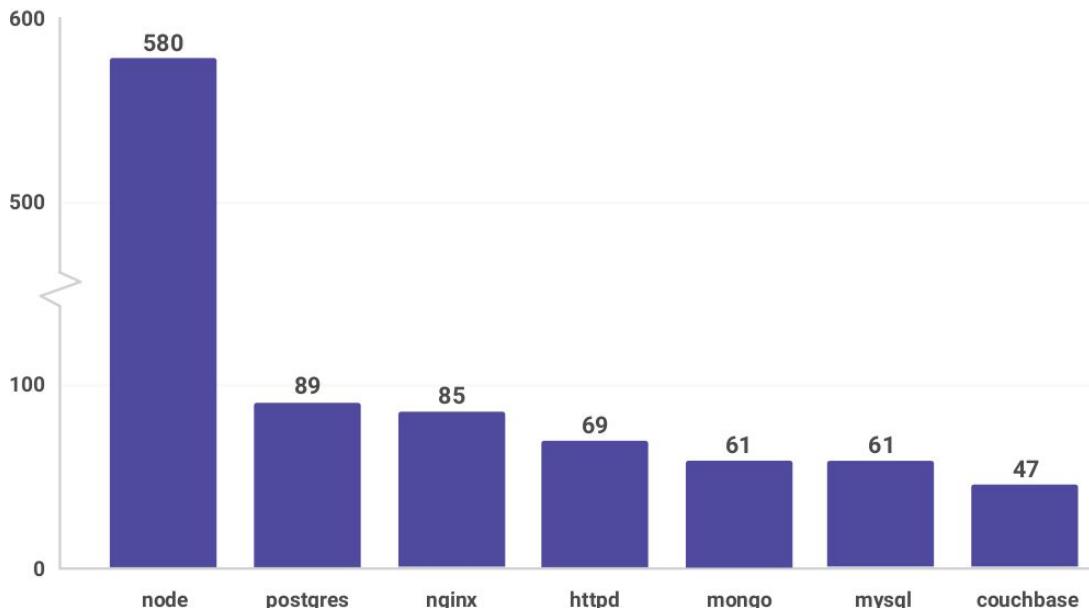
```
docker run -t -i --security-opt=no-new-privileges --cap-drop=ALL  
--cap-add=net_bind_service ubuntu bash
```



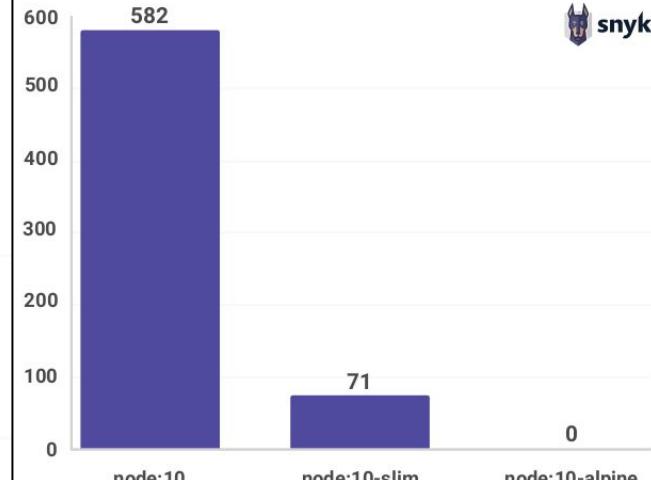
```
docker run -t -i --security-opt=no-new-privileges --cap-drop=ALL  
--cap-add=net_bind_service --cpu-shares=0.5 --memory=200m ubuntu bash
```

# Vulnerable Docker Images

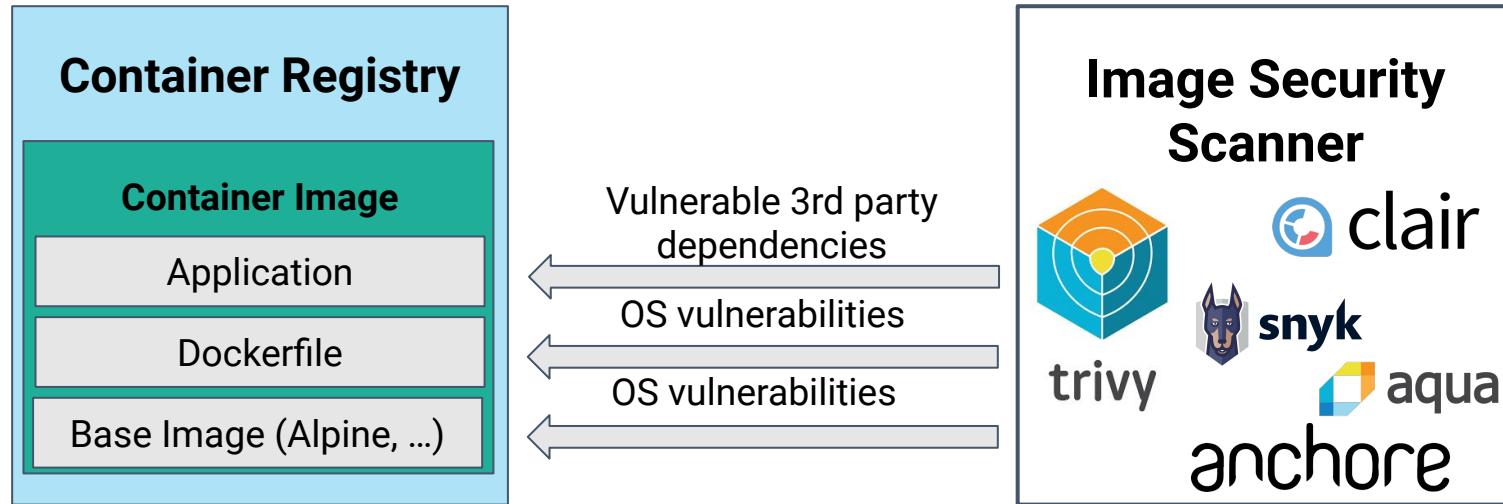
Number of OS vulnerabilities by docker image



Number of vulnerabilities by node image tag



# Container Image Security



<https://anchore.comopensource/>

<https://github.com/coreos/clair>

<https://github.com/aquasecurity/trivy>

<https://www.docker.com/blog/announcing-scanning-from-snyk-for-docker>



# HARBOR

- Open source registry
- Secures artifacts with policies and RBAC
- Ensures images are scanned (Clair or Trivy)
- Signs images as trusted

<https://goharbor.io/>



# Docker and Snyk Announce Partnership to Streamline Container Vulnerability Scanning for Developers

*Millions of developers to benefit from Snyk's vulnerability scanning natively integrated into the Docker workflow for faster and more secure application development*

PALO ALTO, Calif., May 19, 2020 - Docker today announced that it has partnered with Snyk to deliver the first, native vulnerability scanning of container images in Docker. Together, Docker and Snyk will provide a streamlined workflow that makes the application development process more secure for millions of developers, allowing them to more quickly and confidently build secure applications as an automated part of their toolchain.

<https://www.docker.com/press-release/Docker-Snyk-Announce-Partnership-Container-Vulnerability-Scanning>

---

# Live Demo: Show me the code

## Iteration 2: Container Security

<https://github.com/andifalk/secure-development-on-kubernetes>

# Container Security 101

---

- Learn Linux (Security) Basics
- Load Images from Trusted Registries Only
- Scan Images for Vulnerabilities (in CI/CD Pipeline)
- Say No To Root & Run with `--security-opt=no-new-privileges`
- Do NOT hardcode Secrets into a Container Image
- Limit resources (memory, CPU, processes, ...)
- Use Linux Security Module (seccomp, AppArmor, SELinux)

[https://cheatsheetseries.owasp.org/cheatsheets/Docker\\_Security\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Docker_Security_Cheat_Sheet.html)

<https://docs.docker.com/engine/security/>

<https://blog.aquasec.com/docker-security-best-practices>

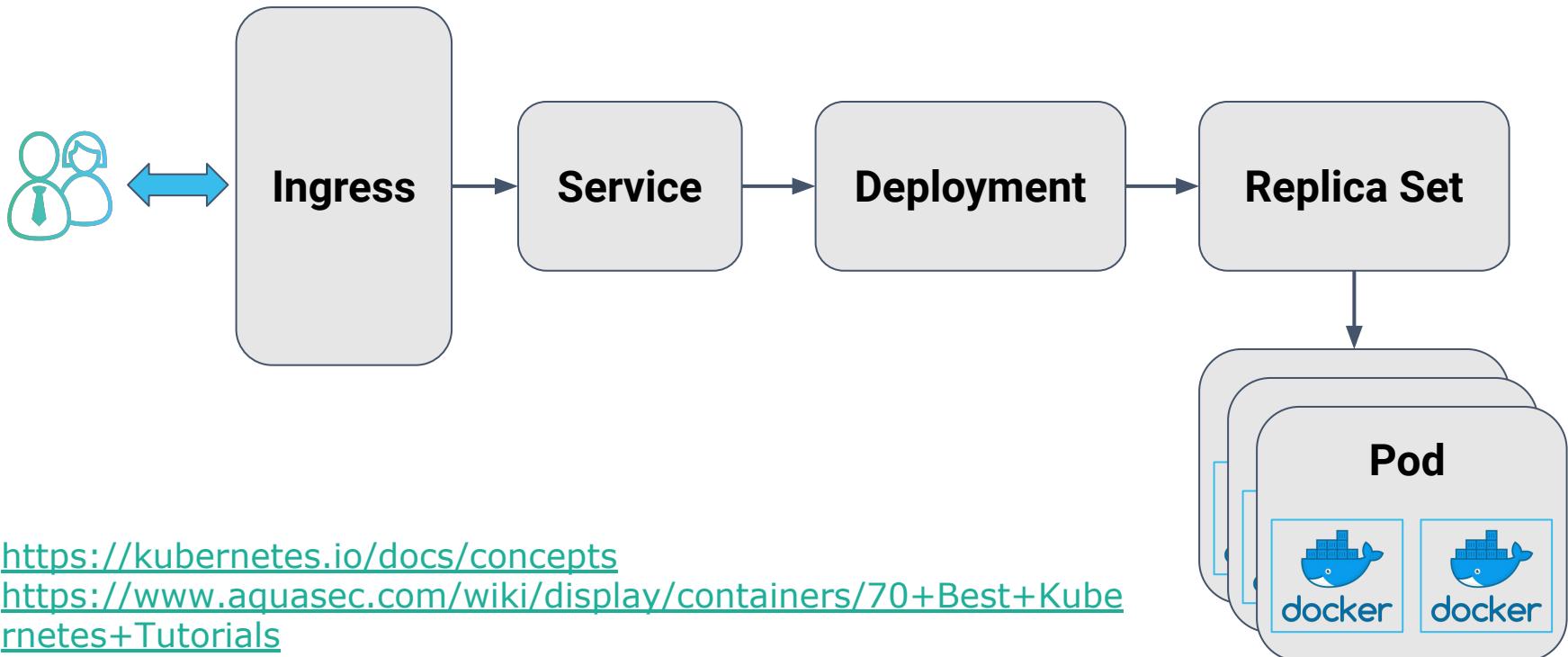
<https://blog.aquasec.com/devsecops-with-trivy-github-actions>

# The Path for Secure Development on K8s

---



# Kubernetes Basics

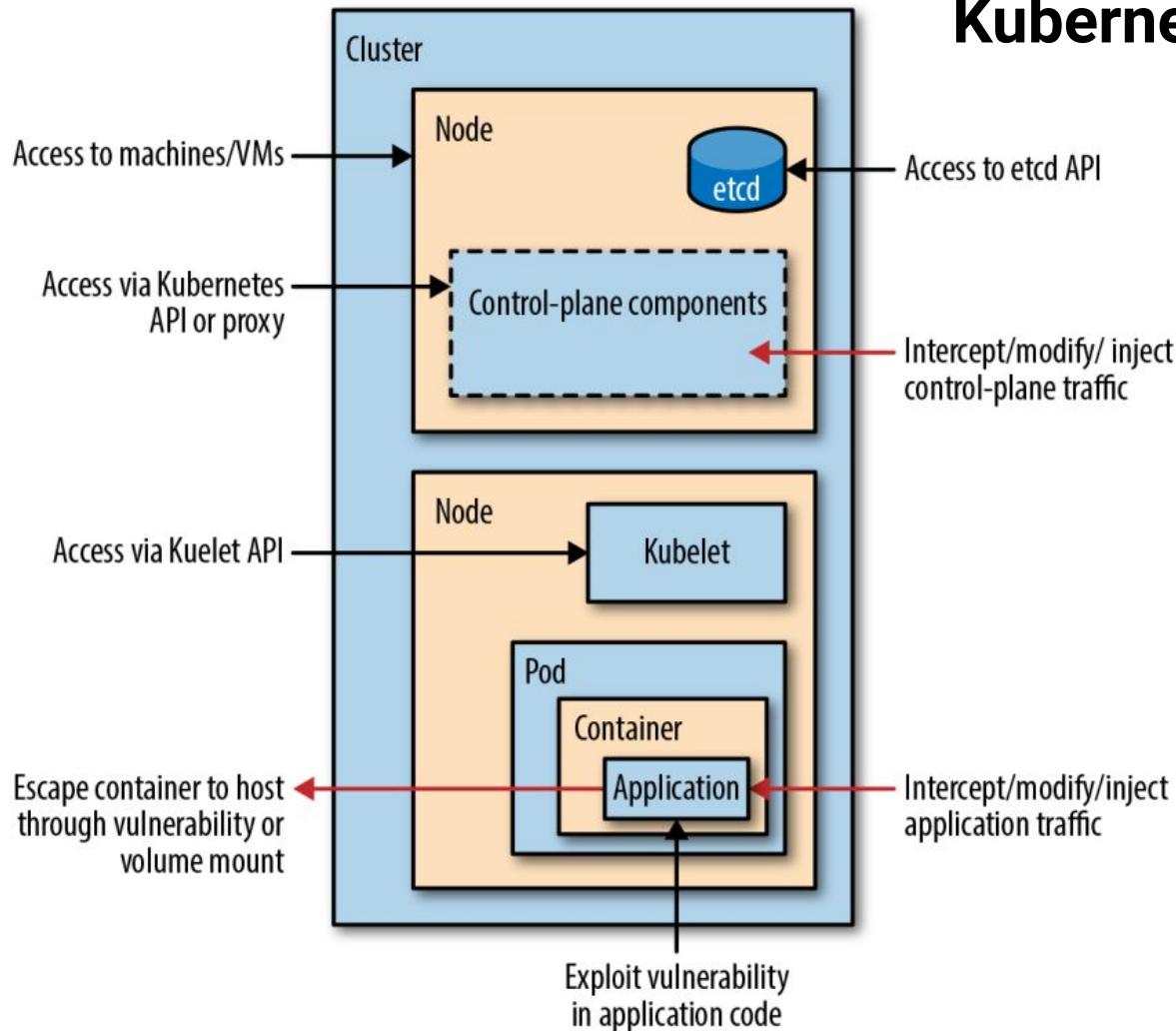


<https://kubernetes.io/docs/concepts>

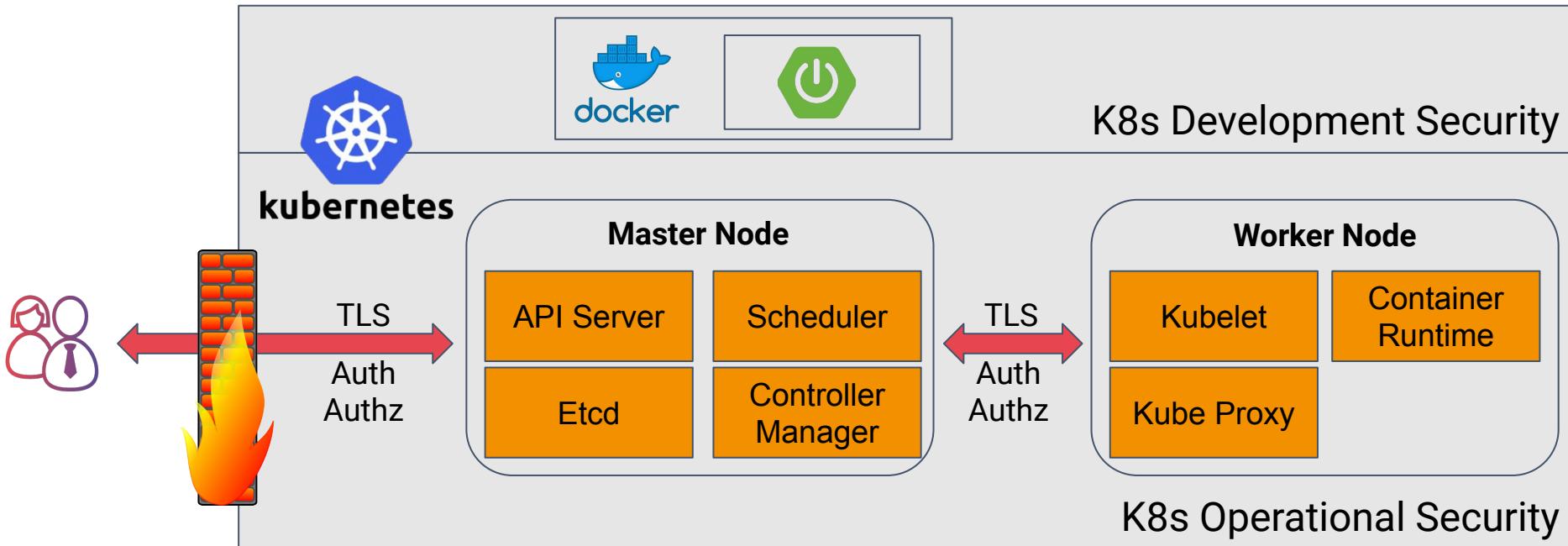
<https://www.aquasec.com/wiki/display/containers/70+Best+Kubernetes+Tutorials>

# Kubernetes attack vectors

Source:  
[Kubernetes Security, O'Reilly, 2018](#)



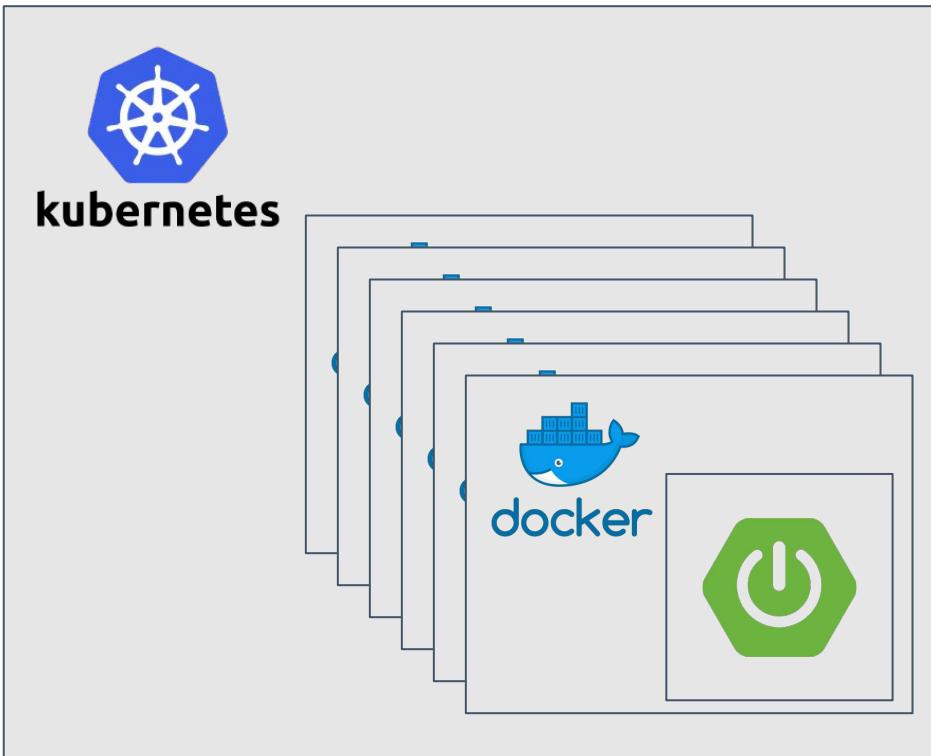
# Operational / Development K8s Security



<https://kubernetes.io/docs/concepts/security/overview/#the-4c-s-of-cloud-native-security>

<https://learnk8s.io/production-best-practices/>

# Kubernetes Security



- Kubernetes Auditing
- Authentication
- Authorization (RBAC)
- Resource Limits
- Pod Security Context
- Pod Security Policy
- Open Policy Agent
- Network Policies

# Kubernetes Authentication (1)

---

- Categories of users:
  - Service accounts managed by Kubernetes
  - Normal users

<https://kubernetes.io/docs/reference/access-authn-authz/authentication/>

# Kubernetes Authentication (2)

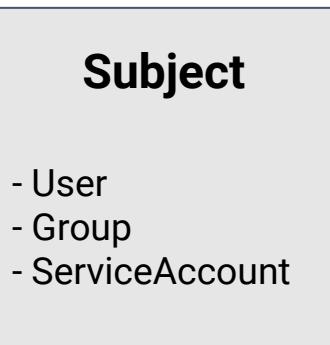
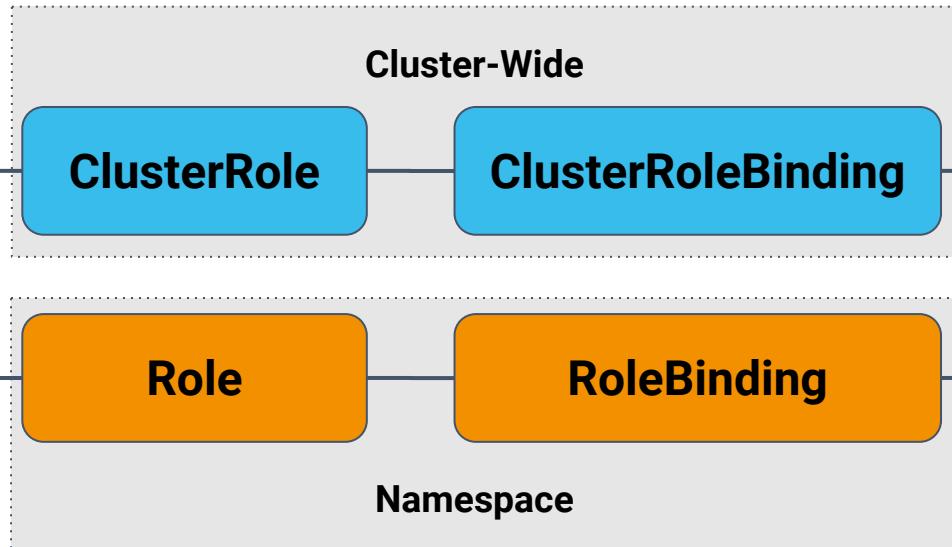
---

- Authentication strategies
  - X509 Client Certs
  - Static Token File
  - Bootstrap Tokens
  - Service Account Tokens
  - OpenID Connect Tokens
  - Webhook Token Authentication
  - Authenticating Proxy

<https://kubernetes.io/docs/reference/access-authn-authz/authentication/>

# Kubernetes Authorization (Role Based Access Control)

Target-Resource(s)



<https://kubernetes.io/docs/reference/access-authn-authz/rbac/>

# Resource Limits

```
spec:  
  ...  
  containers:  
    resources:  
      limits:  
        cpu: "1"  
        memory: "512Mi"  
      requests:  
        cpu: 500m  
        memory: "256Mi"  
  ...
```

<https://kubernetes.io/docs/tasks/configure-pod-container/assign-cpu-resource>

<https://kubernetes.io/docs/tasks/configure-pod-container/assign-memory-resource>

# Pod/Container Security Context

```
spec:  
  securityContext:  
    runAsNonRoot: true  
  containers:  
    securityContext:  
      allowPrivilegeEscalation: false  
      privileged: false  
      runAsNonRoot: true  
      readOnlyRootFilesystem: true  
    capabilities:  
      drop:  
        - ALL
```

<https://kubernetes.io/docs/tasks/configure-pod-container/security-context>

# Pod Security Policy

---

- Policies
  - Privileged
  - Baseline
  - Restricted

<https://kubernetes.io/docs/concepts/security/pod-security-standards/#policies>

# Pod Security Policy (Still In Beta!)

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: no-root-policy
spec:
  privileged: false
  allowPrivilegeEscalation: false
  requiredDropCapabilities:
    - ALL
  runAsUser:
    rule: 'MustRunAsNonRoot'
  ...
```

<https://kubernetes.io/docs/concepts/policy/pod-security-policy>

# Pod Security Policy (Policy Order)

Policy order selection criteria:

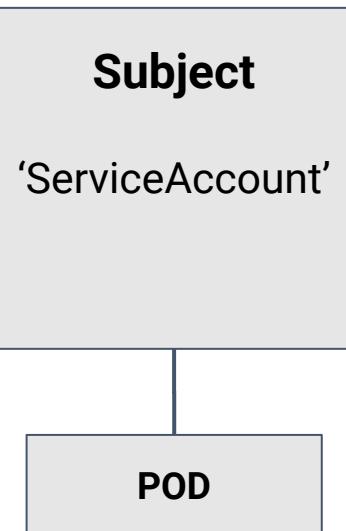
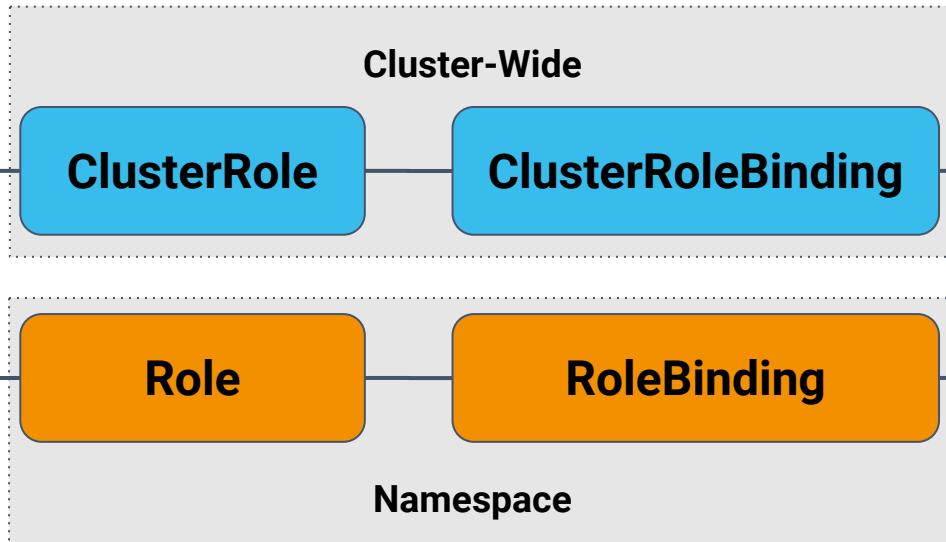
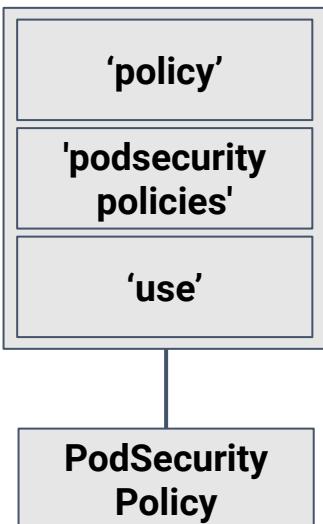
1. Policies which allow the pod as-is are preferred
2. If pod must be defaulted or mutated, the first policy (ordered by name) to allow the pod is selected.

<https://kubernetes.io/docs/concepts/policy/pod-security-policy/#policy-order>

<https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers>

# Kubernetes RBAC + Pod Security Policies

Target-Resource(s)



<https://kubernetes.io/docs/reference/access-authn-authz/rbac/>  
<https://kubernetes.io/docs/concepts/policy/pod-security-policy>

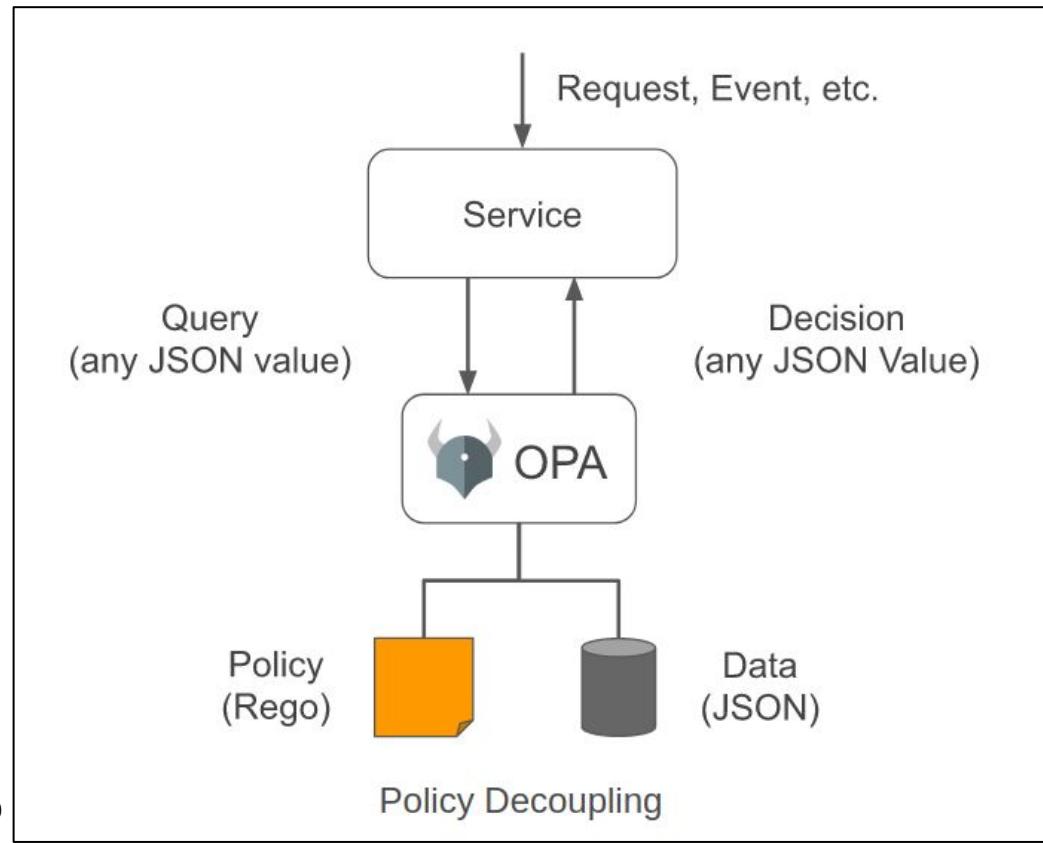
# Kubernetes Role Based Access Control (RBAC)

---

<b>apiGroups</b>	extensions, apps, policy, ...
<b>resources</b>	pods, deployments, configmaps, secrets, nodes, services, endpoints, podsecuritypolicies, ...
<b>verbs</b>	get, list, watch, create, update, patch, delete, use, ...

<https://kubernetes.io/docs/reference/access-authn-authz/rbac/>

# Open Policy Agent



The Rego Playground

```
1 package play
2
3 # Welcome to the Rego playground! Re却是
4 #
5 # Try it out:
6 #
7 #   1. Click Evaluate. Note: 'hello' is
8 #   2. Change "world" to "hello" in the
9 #   3. Change "world" to "hello" on line
10 #
11 # Features:
12 #
13 #   Examples browse a collection
14 #   Coverage view the policy stat
15 #   Evaluate execute the policy w
16 #   Publish share your playground
17 #   INPUT edit the JSON value
18 #   (resize) DATA edit the JSON value
19 #   OUTPUT view the result of p
20
21 default hello = false
22
23 hello {
24     m := input.message
25     m == "world"
26 }
```

INPUT

```
1 { "message": "world" }
```

DATA

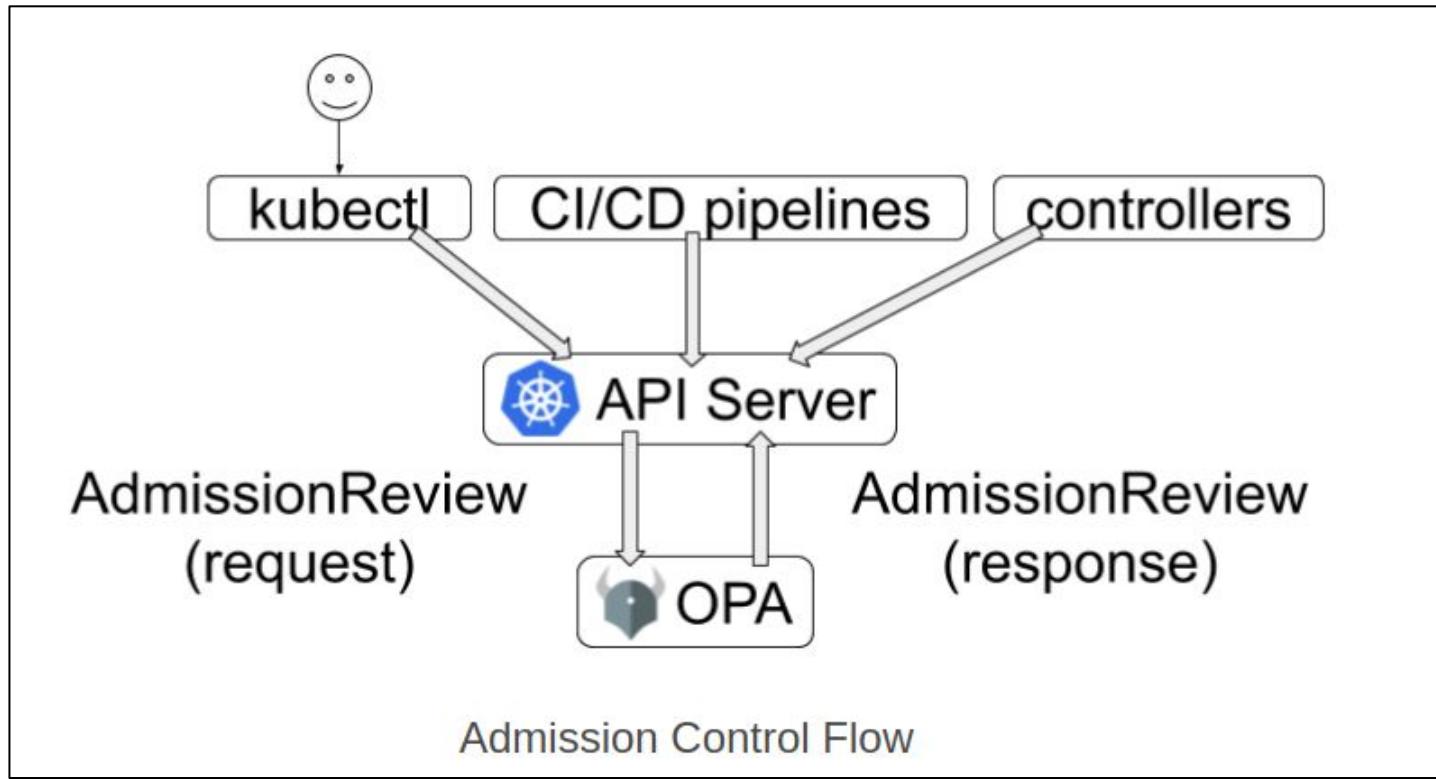
```
Found 1 result in 86.993
```

OUTPUT

```
1 { "hello": true }
```

<https://www.openpolicyagent.org>  
<https://play.openpolicyagent.org>

# Open Policy Agent - Kubernetes Gatekeeper



# Helm 3 Is Here!



Ian Coldwater

@IanColdwater



Folge ich



For people who don't pay attention to the Kubernetes ecosystem: Helm 3.0 is a big deal, removing Tiller and drastically improving the security of that project. Great work, y'all!

<https://v3.helm.sh>

<https://helm.sh/docs/faq/#removal-of-tiller>

---

# Live Demo: Show me the code

## Iteration 3: Kubernetes Security

<https://github.com/andifalk/secure-development-on-kubernetes>

# Kubernetes Security 101

---

- Follow Container Security 101
- Use a Managed Kubernetes Cluster
- Enable Audit Logs
- Enforce Authentication & Role Based Access Control
- Use Pod Security Policies / Open Policy Agent
- Upgrade to Helm Version 3.x (Remove Tiller)
- Monitor your Kubernetes Cluster

<https://cheatsheetseries.owasp.org>

<https://owasp.org/www-project-top-ten>

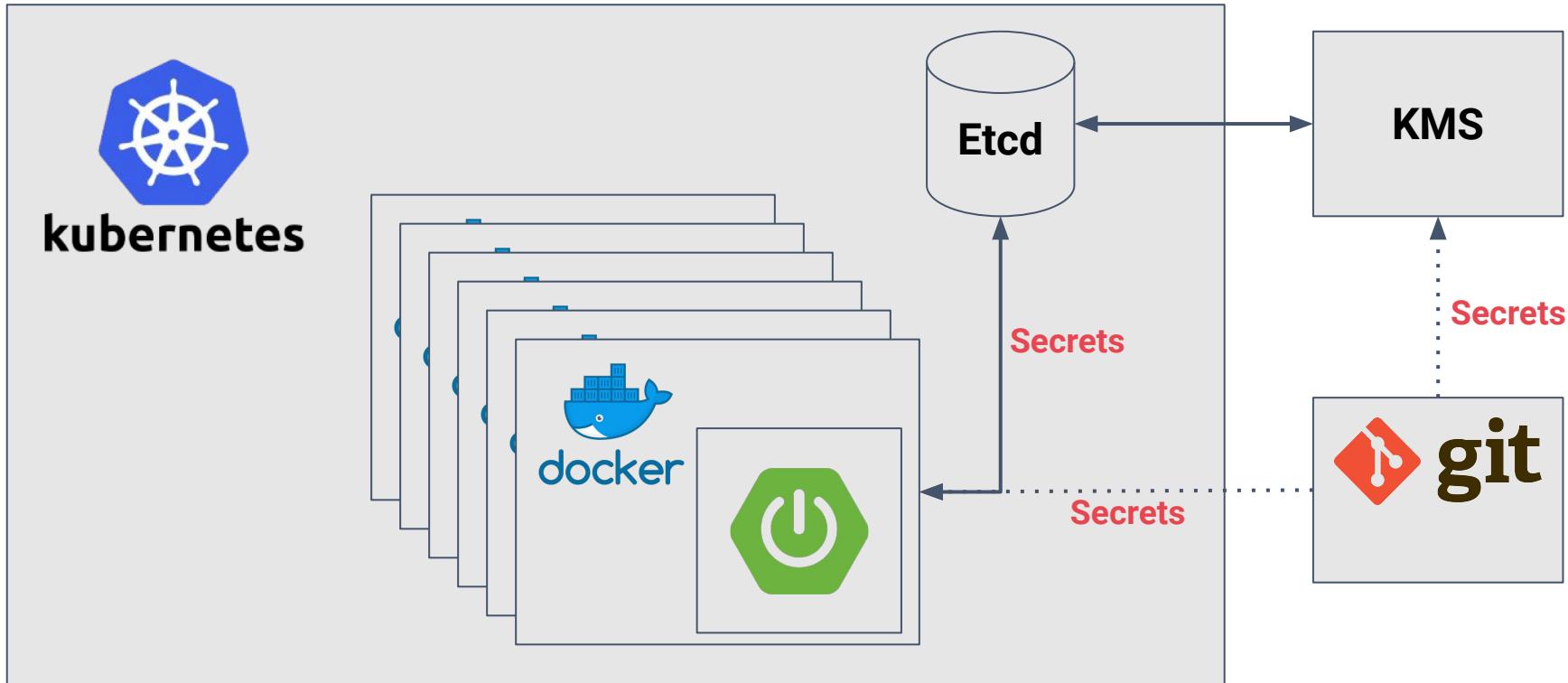
<https://owasp.org/www-project-proactive-controls>

# The Path for Secure Development on K8s

---



# Kubernetes Secrets



# Kubernetes Secrets

```
apiVersion: v1
kind: Secret
metadata:
  name: hello-spring-cloud-kubernetes
  namespace: default
type: Opaque
data:
  user.username: dXNlcg==
  user.password: azhzX3VzZXI=
  admin.username: YWRtaW4=
  admin.password: azhzX2FkbWlu
```

<https://kubernetes.io/docs/concepts/configuration/secret>

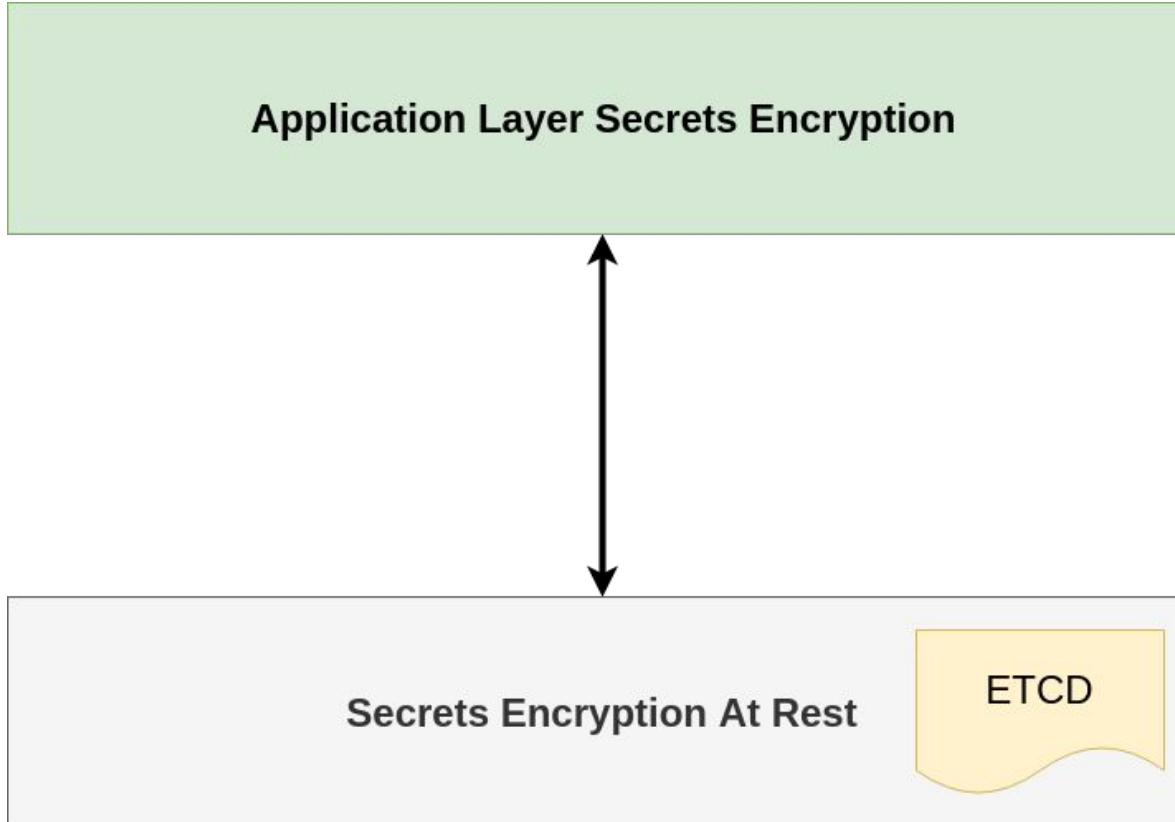
# Pay Attention to Spring Boot Actuator

```
{  
    "name": "applicationConfig: ...",  
    "properties": {  
        "greet.my-sec": {  
            "value": "geheim",  
            "origin": "class path resource ..."  
        },  
        "greet.password": {  
            "value": "*****",  
            "origin": "class path resource ..."  
        }  
    }  
}
```

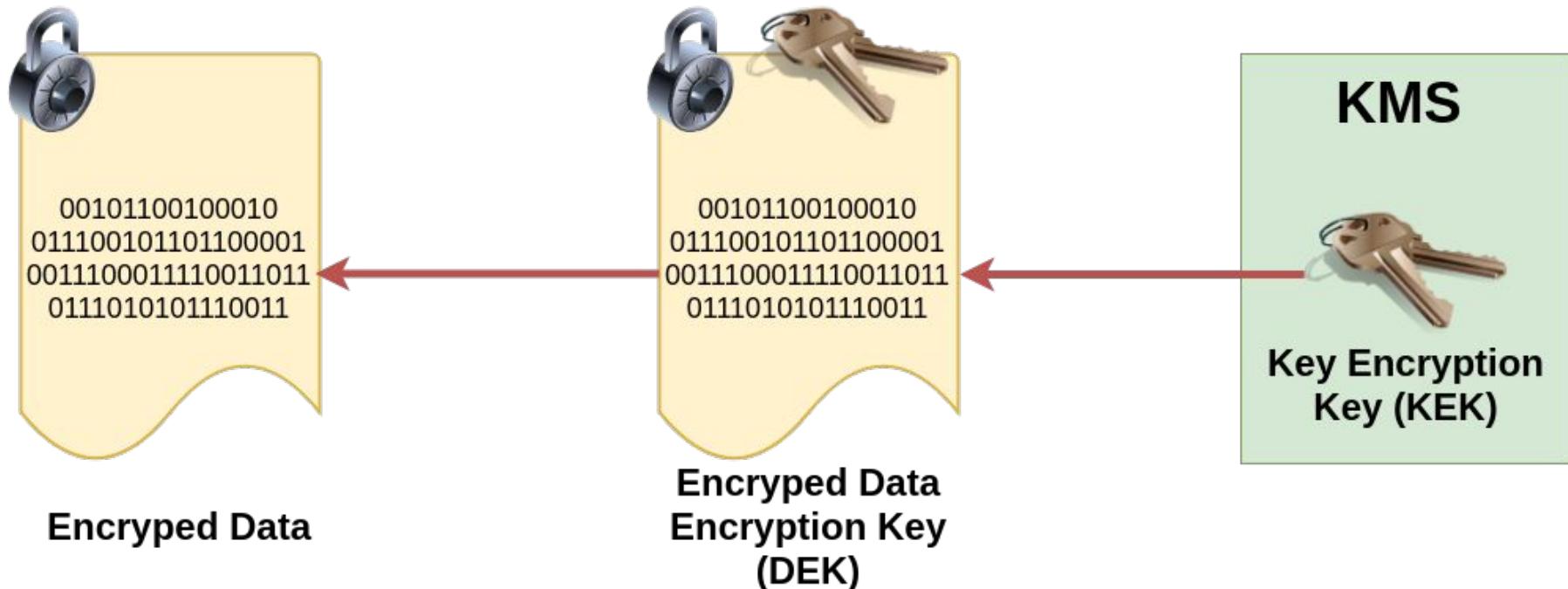
<http://localhost:8080/actuator/env>

# Encryption Layers

---



# Envelope Encryption On Kubernetes



<https://cloud.google.com/kms/docs/envelope-encryption>

<https://kubernetes.io/docs/tasks/administer-cluster/kms-provider>

# Key Management System (KMS) Providers

---

- Azure Key Vault
- Google Cloud KMS
- AWS KMS
- Hashicorp Vault
- ...

<https://github.com/Azure/kubernetes-kms>

<https://github.com/Azure/kubernetes-keyvault-flexvol>

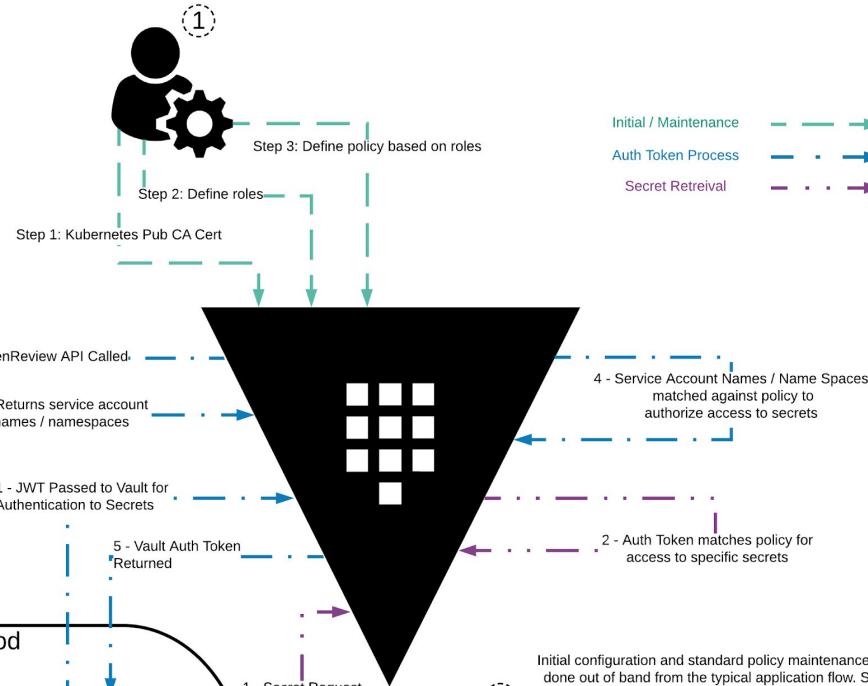
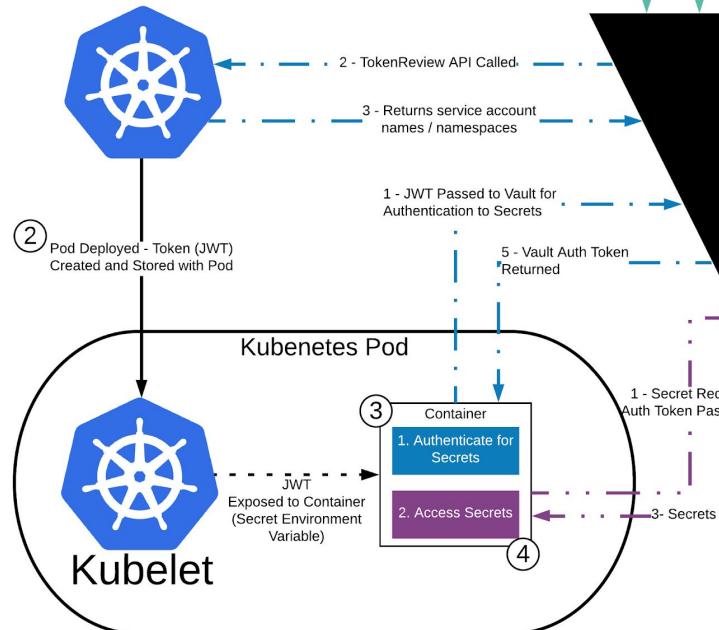
<https://cloud.google.com/kms>

<https://aws.amazon.com/de/kms>

<https://learn.hashicorp.com/vault/kubernetes/external-vault>

# Kubernetes Secret Authentication and Access with Vault

## Kubectl



Initial configuration and standard policy maintenance is done out of band from the typical application flow. So consider this a one time setup / infrequent action.  
 If Policy as Code is implemented, this may be an exception, as creating roles and policies can happen as part of a deployment process as well (steps 2 and/or 3).

(1) Pod deployed automatically, or by human intervention. The JWT is automatically included as part of the secret store.

(2) Single API call to Vault to authenticate. Vault handles the rest of the calls and determines which policies should be attached to the Auth Token. This token is returned to the client to be utilized for all future requests for secrets, as long as the lease is valid.

(3) Single API call to Vault, including the Auth Token. Vault returns appropriate secrets.

# What about Secrets in git

---

- Sealed Secrets
- Helm Secrets
- Kamus
- Sops
- Hashicorp Vault

<https://learnk8s.io/kubernetes-secrets-in-git>

<https://github.com/bitnami-labs/sealed-secrets>

<https://github.com/futuresimple/helm-secrets>

<https://github.com/Soluto/kamus>

<https://github.com/mozilla/sops>

<https://www.vaultproject.io>

# Kubernetes Secrets - Best Practices

---

- Encrypt Secret Data at Rest & in Transit
  - Only Base64 encoded by default in Etcd!
- Restrict interactions with secrets API (RBAC)
- Mount secrets instead of ENV Mapping

<https://kubernetes.io/docs/concepts/configuration/secret/#best-practices>  
<https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data>

---

# Summary

## Summary / Key Insights

---

- Kubernetes is Complex - Check Alternatives!
- Follow Application Security 101
- Follow Container & Kubernetes Security 101
- Ensure your secrets are encrypted in K8s
- Never store secrets in Source Control (Git, ...)
- Check out the Demos:  
<https://github.com/andifalk/secure-development-on-kubernetes>

---

# Books and Online References

# Books and Online References (1)

---

- [Kubernetes Security, O'Reilly, 2018, ISBN: 978-1-492-04600-4](#)
- [Container Security, O'Reilly, 2020, ISBN: 978-1492056706](#)
- [`https://github.com/andifalk/secure-development-on-kubernetes`](#)
- [Crafty Requests: Deep Dive Into Kubernetes CVE-2018-1002105 - Ian Coldwater \(Video\)](#)
- [Ship of Fools: Shoring Up Kubernetes Security - Ian Coldwater \(Video\)](#)
- [`https://kubernetes.io/docs/concepts/security/overview/#the-4c-s-of-cloud-native-security`](#)
- [`https://kubernetes.io/docs/tasks/administer-cluster/securing-a-cluster`](#)
- [`https://opensource.com/article/18/3/just-say-no-root-containers`](#)
- [`https://github.com/GoogleContainerTools/jib`](#)
- [`https://anchore.com/opensource/`](#)
- [`https://github.com/coreos/clair`](#)
- [`https://github.com/aquasecurity/trivy`](#)
- [`https://www.owasp.org/index.php/OWASP\_Docker\_Top\_10`](#)

# Books and Online References (2)

---

- <https://kubernetes.io/docs/tasks/configure-pod-container/assign-cpu-resource>
- <https://kubernetes.io/docs/tasks/configure-pod-container/assign-memory-resource>
- <https://kubernetes.io/docs/tasks/configure-pod-container/security-context>
- <https://kubernetes.io/docs/concepts/policy/pod-security-policy>
- <https://kubernetes.io/docs/reference/access-authn-authz/rbac/>
- <https://kubernetes.io/docs/concepts/configuration/secret>
- <https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data>
- <https://cloud.google.com/kms/docs/envelope-encryption>
- <https://kubernetes.io/docs/tasks/administer-cluster/kms-provider>
- <https://github.com/Azure/kubernetes-kms>
- <https://cloud.google.com/kms>
- <https://aws.amazon.com/de/kms>

---

**Thank You very much!  
Questions?**