# kdigger

## Kubernetes focused container assessment and context discovery tool for penetration testing

Mahé Tardy (@mtardy_)

Quarkslab

# About me

- **Mahé Tardy**
- Security R&D Engineer @ **Quarkslab**
- Doing research on **Kubernetes**
- **@mtardy_**
- mtardy@quarkslab.com

Quarkslab

# CVEs of March for container escapes

## Linux

- **CVE-2022-0847** a.k.a. DirtyPipe. Vulnerability allows for overwrite of files that should be read-only.

- **CVE-2022-0492**. Vulnerability in cgroup handling can allow for container breakout depending on isolation layers in place.

## CRI-O

- **CVE-2022-0811.** Vulnerability in setting sysctls in k8s/ OpenShift manifests allows for container breakout.

## Containerd

- **CVE-2022-23648**. Vulnerability in volume mounting allows for arbitrary file read from the underlying host, leading to likely indirect container breakout.

*Source: [https://www.container-security.site/](https://www.container-security.site/) by @raesene*

# We don’t need that!

## Linux

- **CVE-202** . DirtyPipe. V ws for overwrite of uld be re

- **CVE-2022-0** bilit dling can allow for container b on layers in place.

## CRI-O

- **CVE-2022-0811.** tting sysctls in k8s/ OpenShift manifes ner breakout.

## Containerd

- **CVE-202** mounting allows for arbitr leading to likely break

*Source* *container-secur* *by @raesene*

**Kubernetes security** today is more about **configuration** than **vulnerabilities.**

Quarkslab

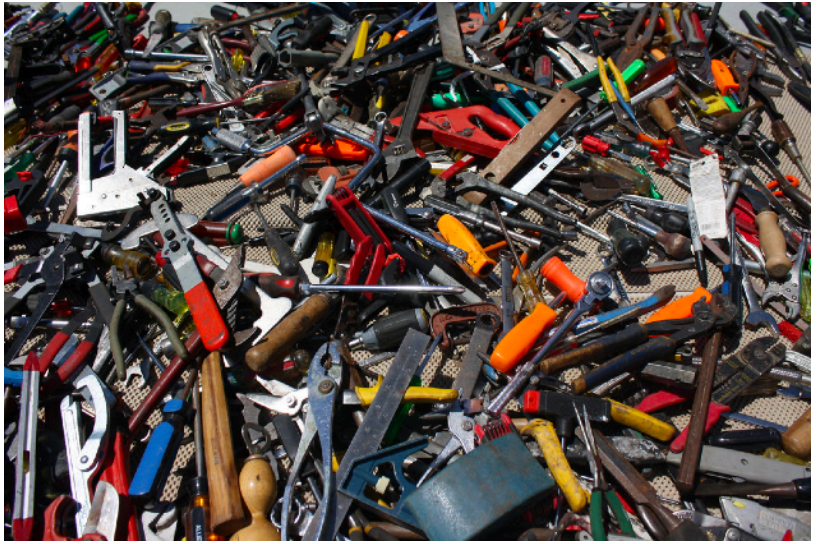
# Why another tool?

Participated in the last 4 KubeCon Cloud-Native Security Day CTFs.

Learned the habits of many experts in the fields.

Loved **amicontained** by Jessie Frazelle.

Decided to automate a security checklist from inside a Kubernetes Pod!



# What can kdigger do for you?

*Kubernetes focused container assessment and context discovery tool for penetration testing*

Like **amicontained** you can:

- Try to guess your container runtime.
- See your capabilities.
- Scan for namespace activation and configuration.
- Scan for the allowed syscalls.

And more **basic** stuff: check mounts, uid, processes, devices, status flag, and API versions.

But from a **Kubernetes** perspective:

- Retrieve service account token.
- Scan token permissions.
- List interesting environment variables.
- Retrieve all available services in a cluster.
- Retrieve leaked information by cgroups v1.
- Retrieve the specifications of the node.
- Scan the admission controller chain!
- Retrieve all the CRDs

# What can kdigger do for you?

*Kubernetes focused container assessment and context discovery tool for penetration testing*

Like **amicontained** you can:

- Try to guess your container runtime.
- See your capabilities.
- Scan for namespace activation and configuration.
- Scan for the allowed syscalls.

And more **basic** stuff: check mounts, uid, processes, devices, seccus fag, and API versions.

But from a **Kubernetes** perspective:

- Retrieve service account token.
- Scan token permissions.
- List interesting environment variables.
- Retrieve all available services in a cluster.
- Retrieve leaked information by cgroups v1.
- Retrieve the specifications of the node.
- Scan the admission controller chain!
- Retrieve all the CRDs

# Disclaimer on usage

Some checks rely on implementation details or fragile features:

- PID namespace or container runtime check from **amicontained**.
- CoreDNS wildcard feature for services listing has been removed in v1.9.0.
  https://github.com/coredns/coredns/issues/4984

Needs to be updated and extended with new checks

Straightforward to extend, look at the contributing guide in the README if you have interesting checks to add.

# Demonstration!

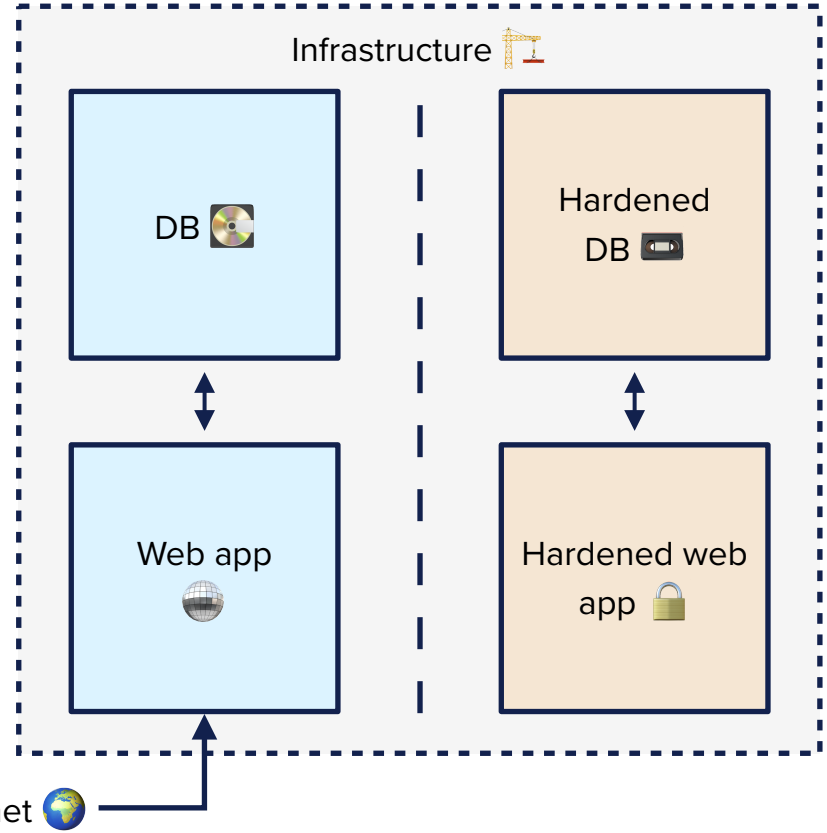
# Simplified infrastructure

This setup was inspired by a mission and highlights the problems of multi tenancy  in clusters.

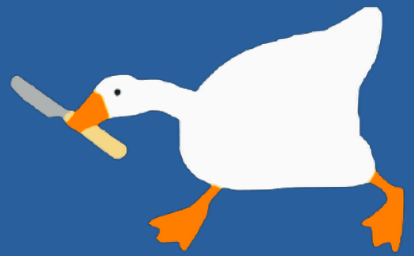
**Context:**

- Web app exposed to internet.

- Hardened web app, only exposed on the internal network.



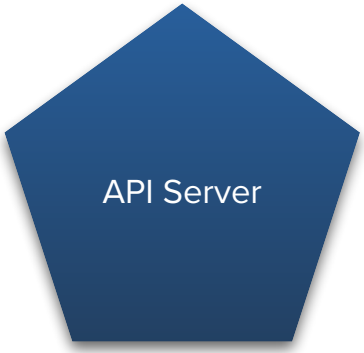
# Honk time!



Quarkslab

# What exactly happened?

# What happened? - Three main issues




Node pool 🐟

Node 1

Namespace default

Web app Pod 🧨

Container 📦

Namespace secure

Hardened DB 📚

Node 2

Namespace other

Pod 😼

Pod 😼

...

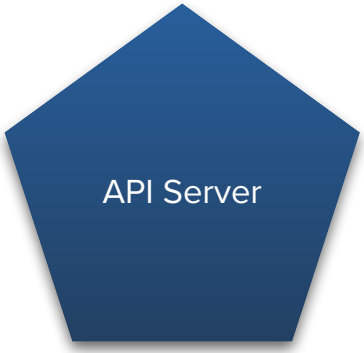
Namespace secure

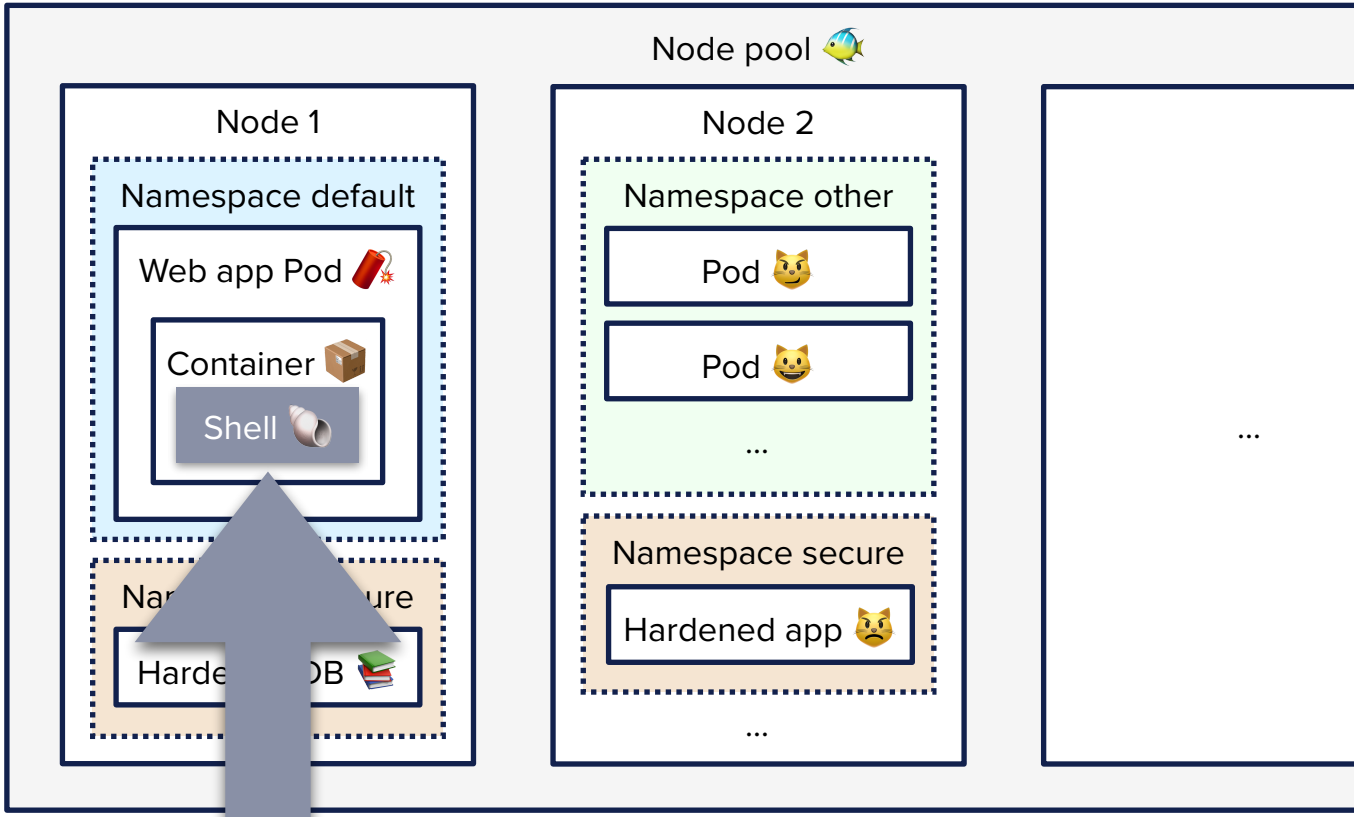
Hardened app 😾

...

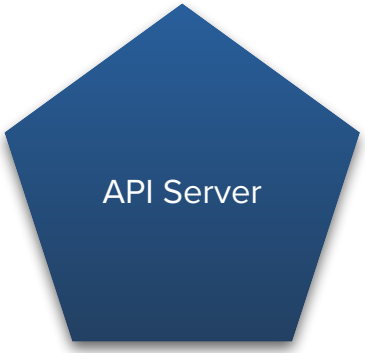
...

# What happened? - First issue: vulnerability in software



# What happened? - Second issue: credentials in container




Node pool

Node 1

Namespace default

Web app Pod

Container

Shell

Namespace secure

Hardened DB

Node 2

Namespace other

Pod

Pod

...

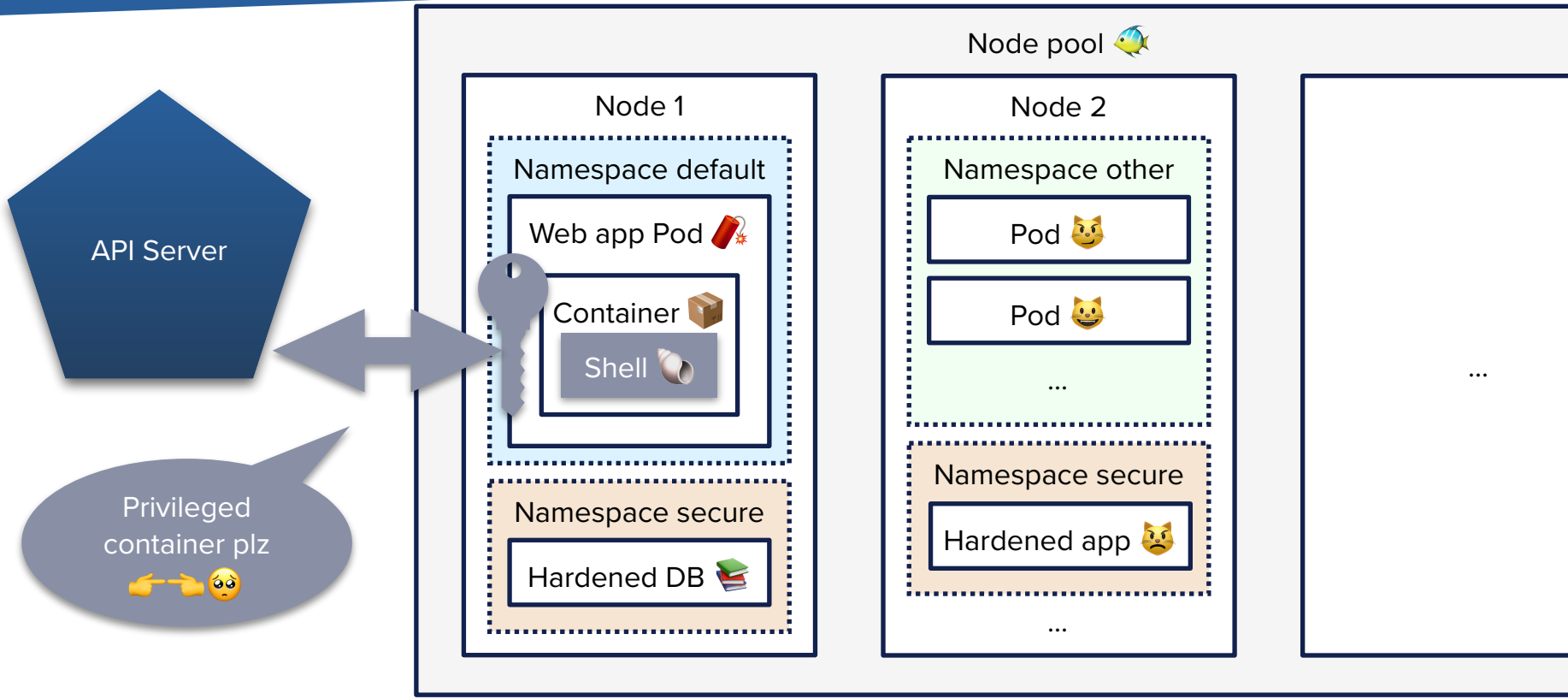
Namespace secure

Hardened app

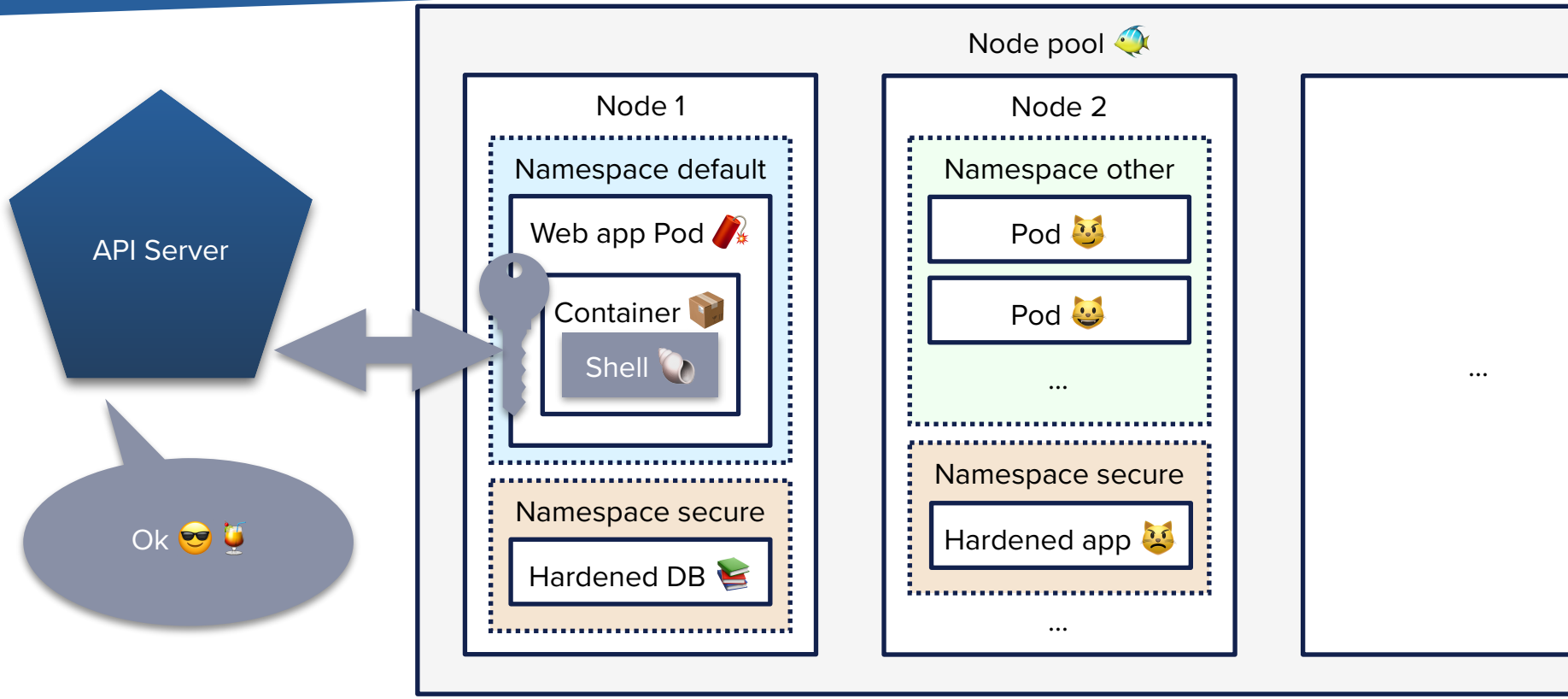
...

...

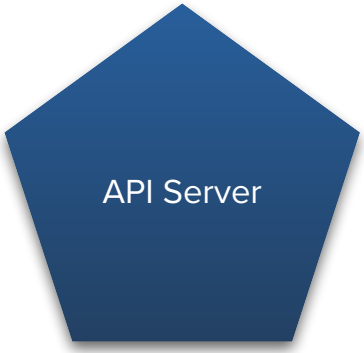
# What happened? - Third issue: no admission control



# What happened? - Third issue: no admission control



# What happened? - Third issue: no admission control




Node pool

Node 1

Namespace default

Web app Pod

Container

Privileged pod

Namespace secure

Hardened DB

Node 2

Namespace other

Pod

Pod

...

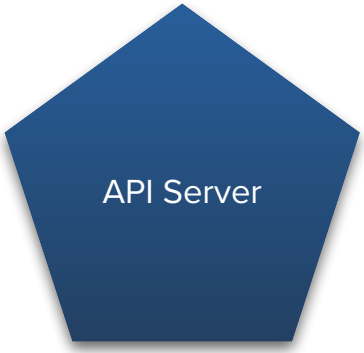
Namespace secure

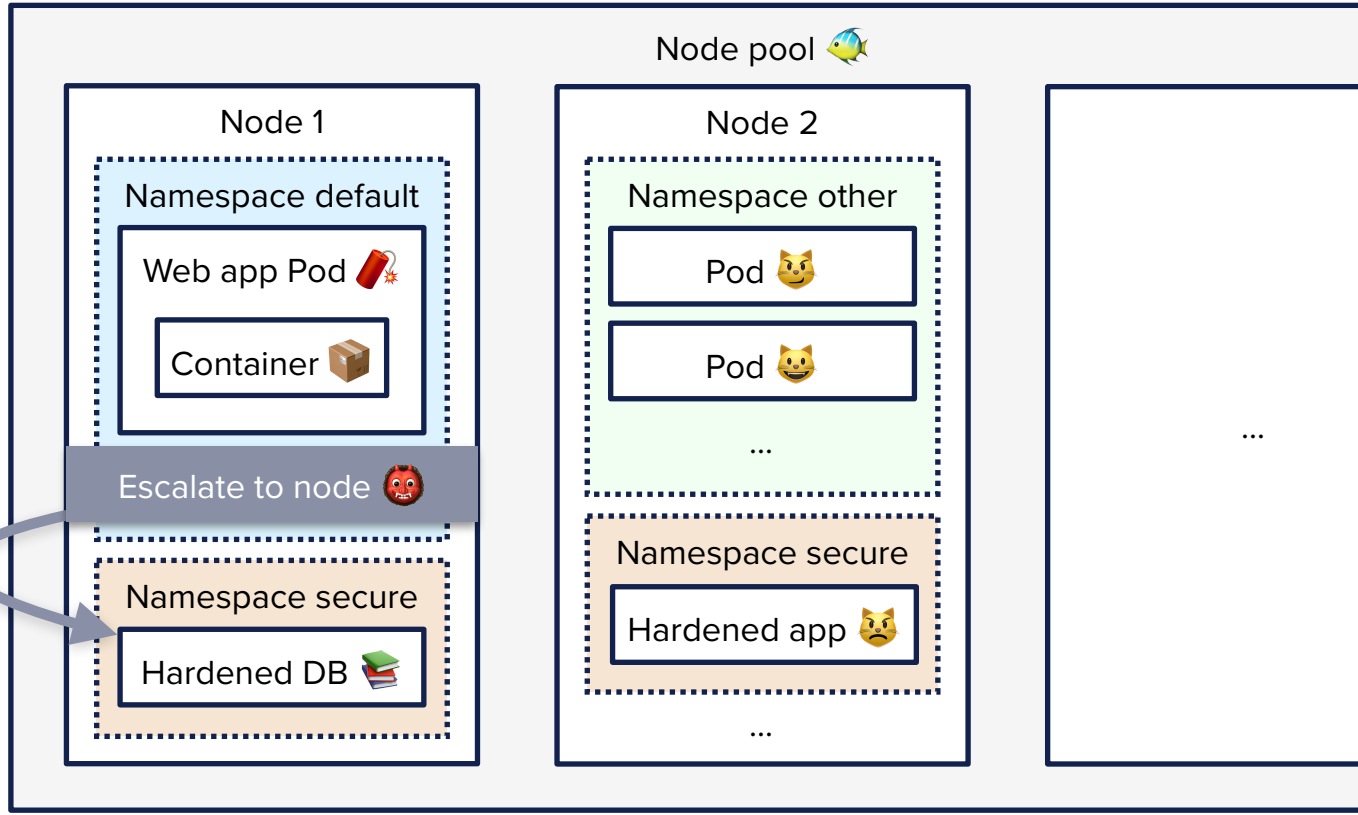
Hardened app

...

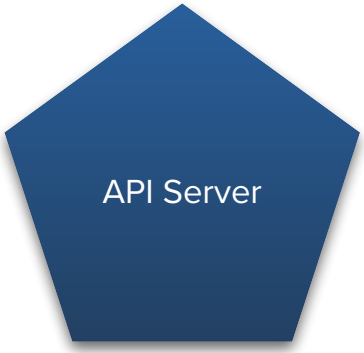
...

# What happened? - Third issue: no admission control



# What happened? - Third issue: no admission control




Node pool

Node 1

Namespace default

Web app Pod

Container

Escalate to node

Namespace secure

Hardened DB

Node 2

Namespace other

Pod

Pod

Escalate to node

Namespace secure

Hardened app

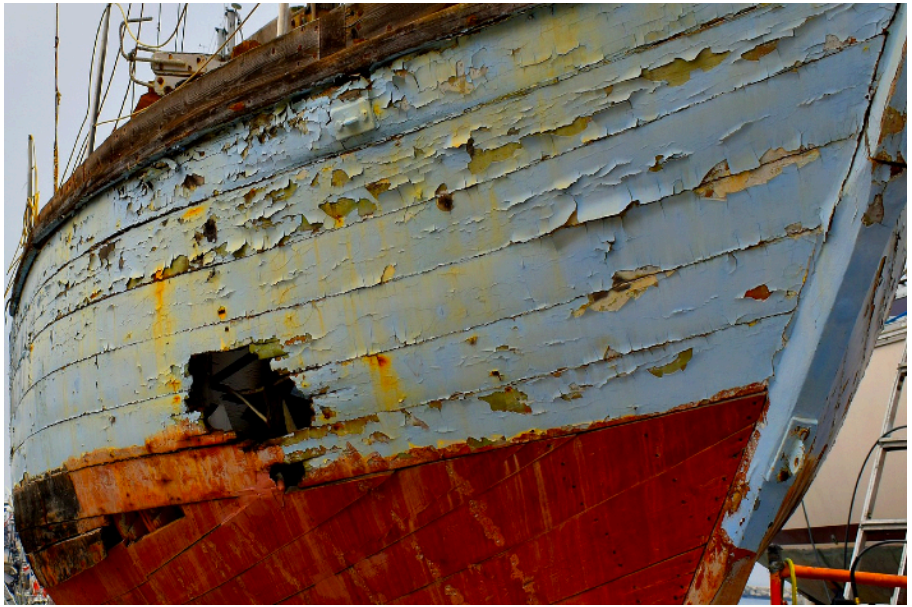
...




...

# How to fix these issues?

# First issue: vulnerabilities in software



**Not related to Kubernetes, some ideas:**

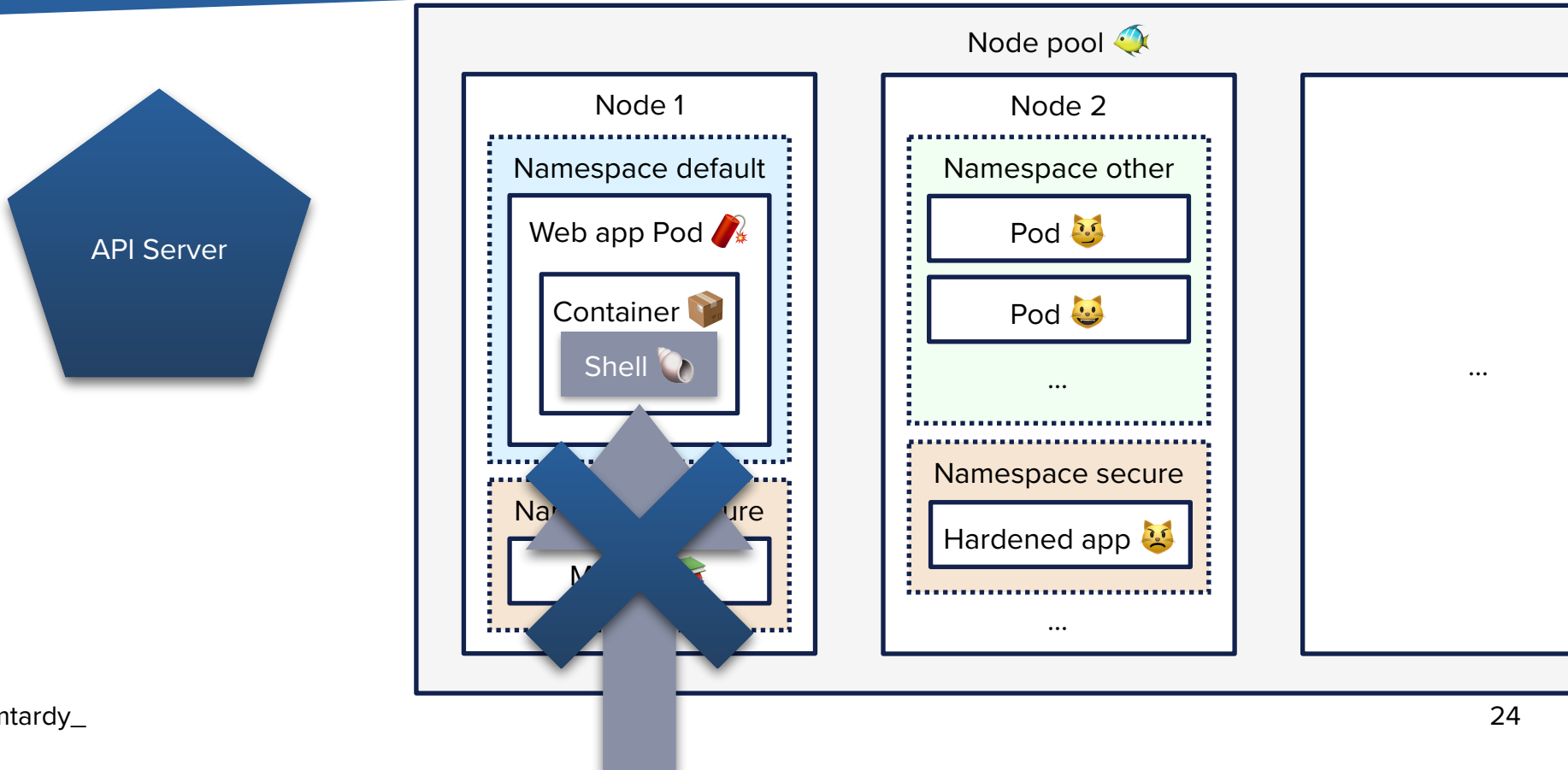
- Write software without vulnerabilities (easy).

- Audit your code by third party (hehe).

- Scan your container images for known vulnerabilities (lot of choices). And truly fix them!

- So many things...

# Solving - First issue: vulnerability in software

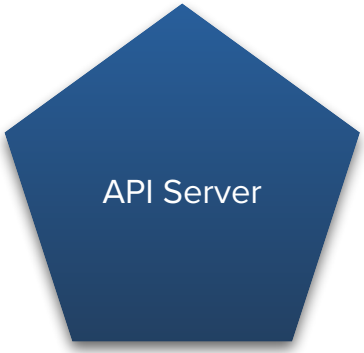


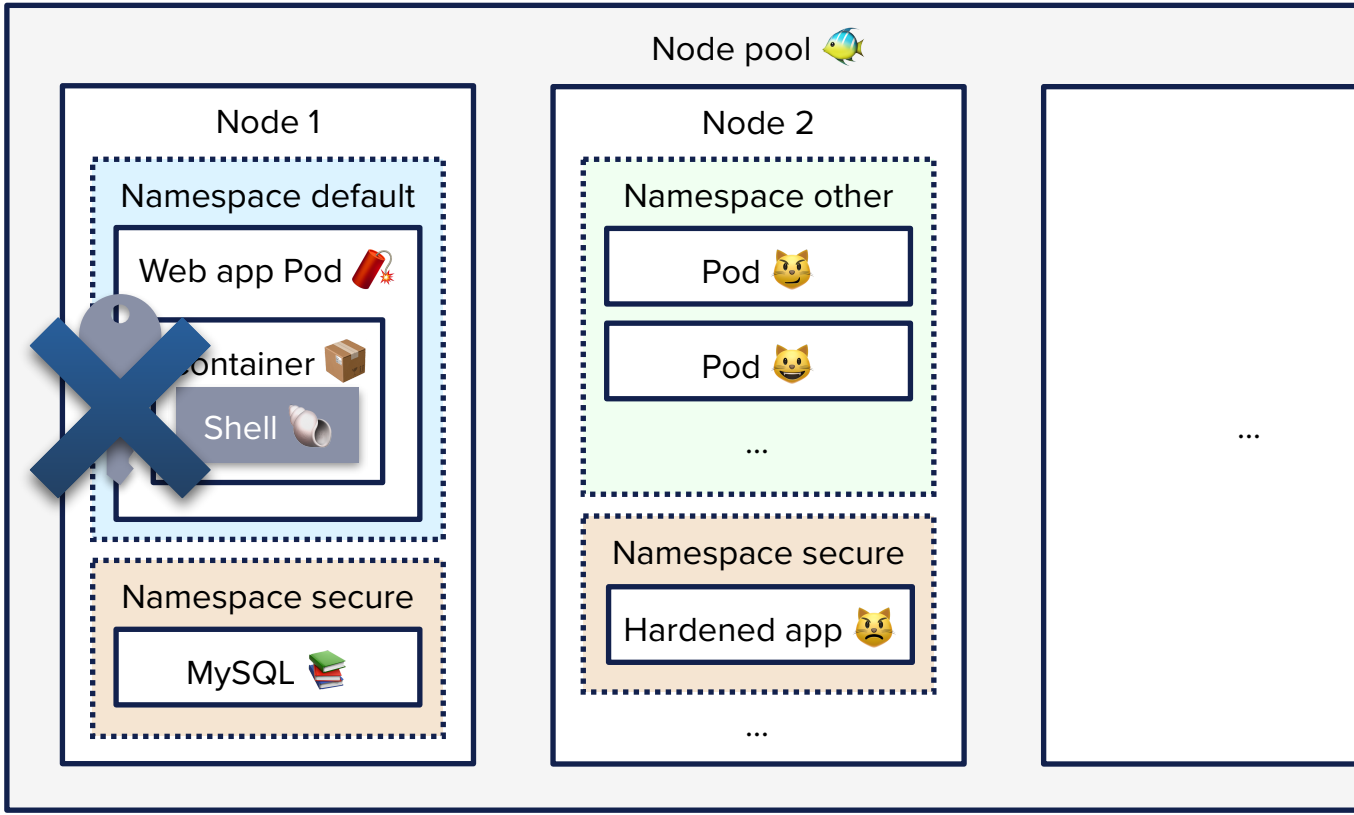
# Second issue: credentials in container


**Be extra careful with rights and tokens**

- Configure wisely your RBAC.
- Avoid roles with a lot of rights.
- Be extra careful with rights on resources like Pods or similar with create or update verbs.
- Avoid mounting service account token if possible.
- See https://kubernetes.io/docs/concepts/security/rbac-good-practices/.

# Solving - Second issue: credentials in container



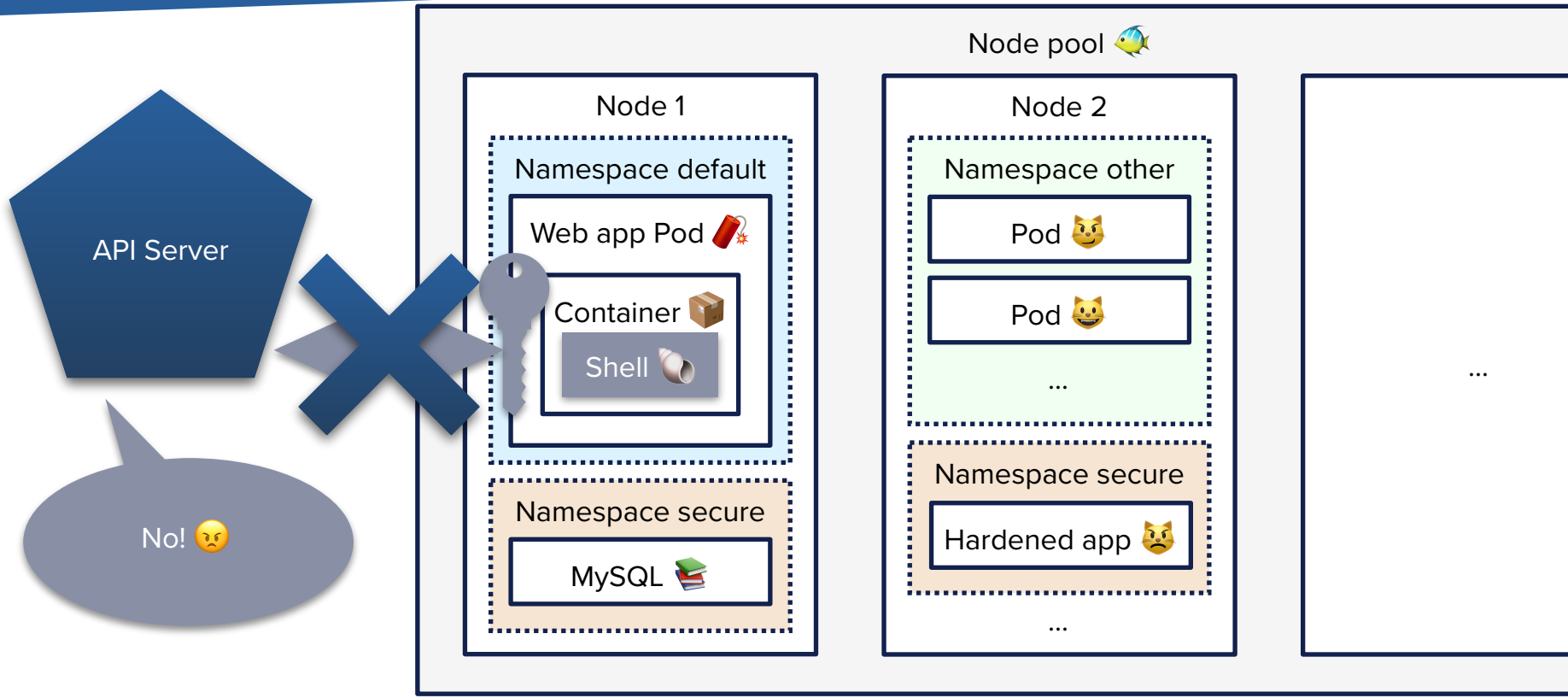



# Third issue: no admission control

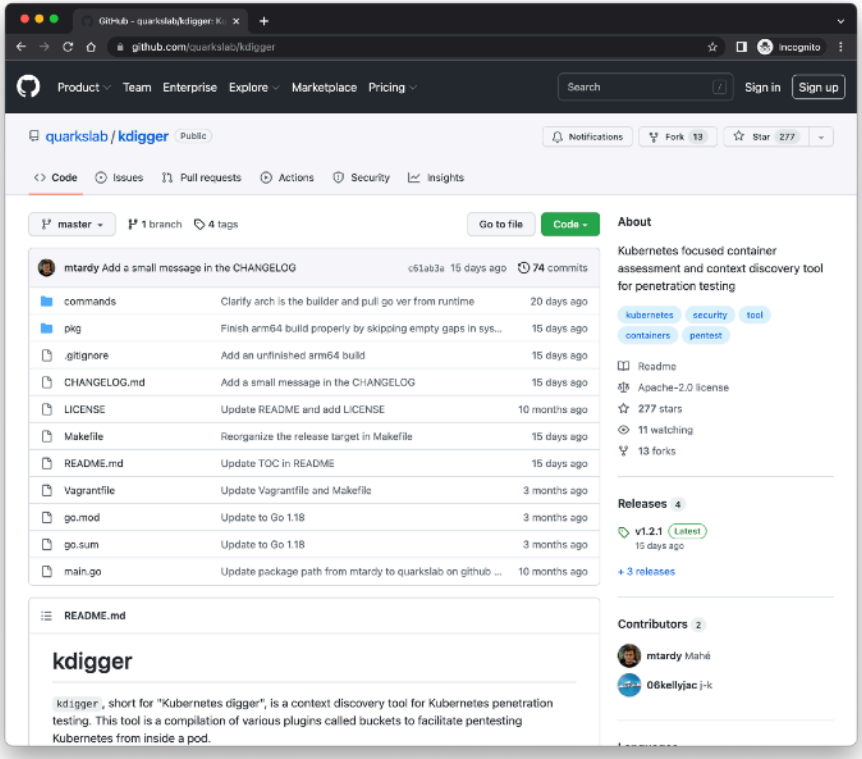
**Kubernetes RBAC is not enough.**

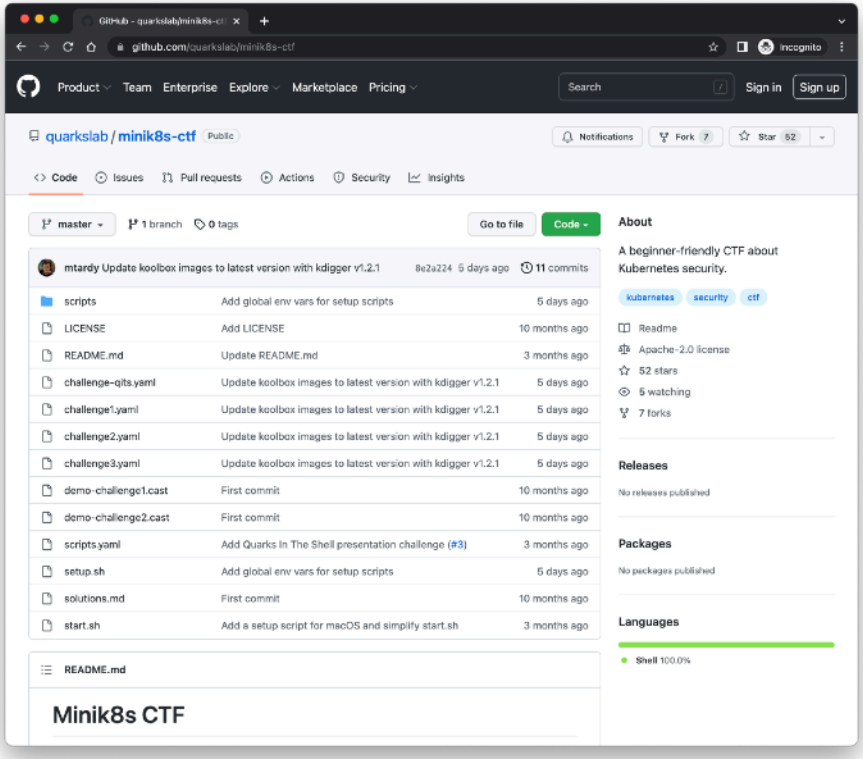
- RBAC is allow/disallow to create pod but this is the maximum granularity.
- Use admission controllers to be more granular.
- Integrated one: **PodSecurityPolicy** deprecated in 1.21 and removed in 1.25. Replacement **PodSecurity** is the new standard, in beta since last version, 1.23.
- Third party ones: Kyverno, OPA/Gatekeeper, etc.

# Solving - Third issue: no admission control

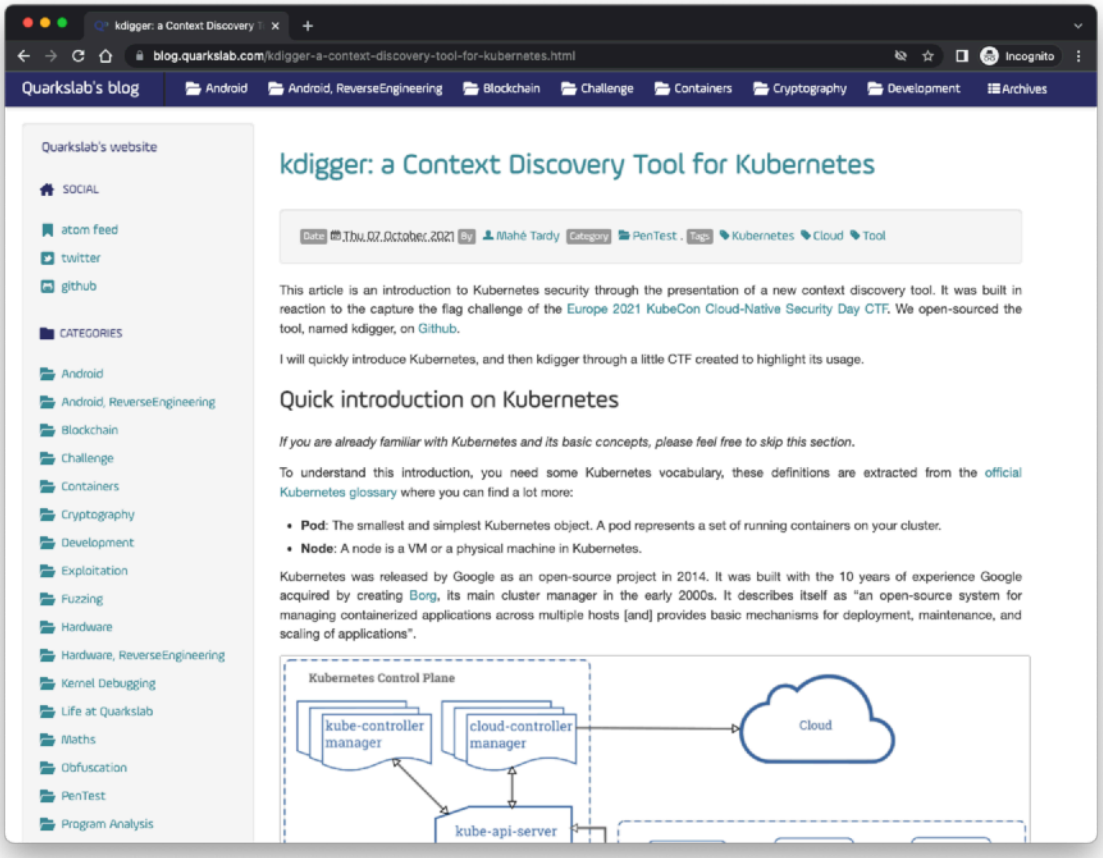


# If you want to experiment by yourself



# If you want to experiment by yourself



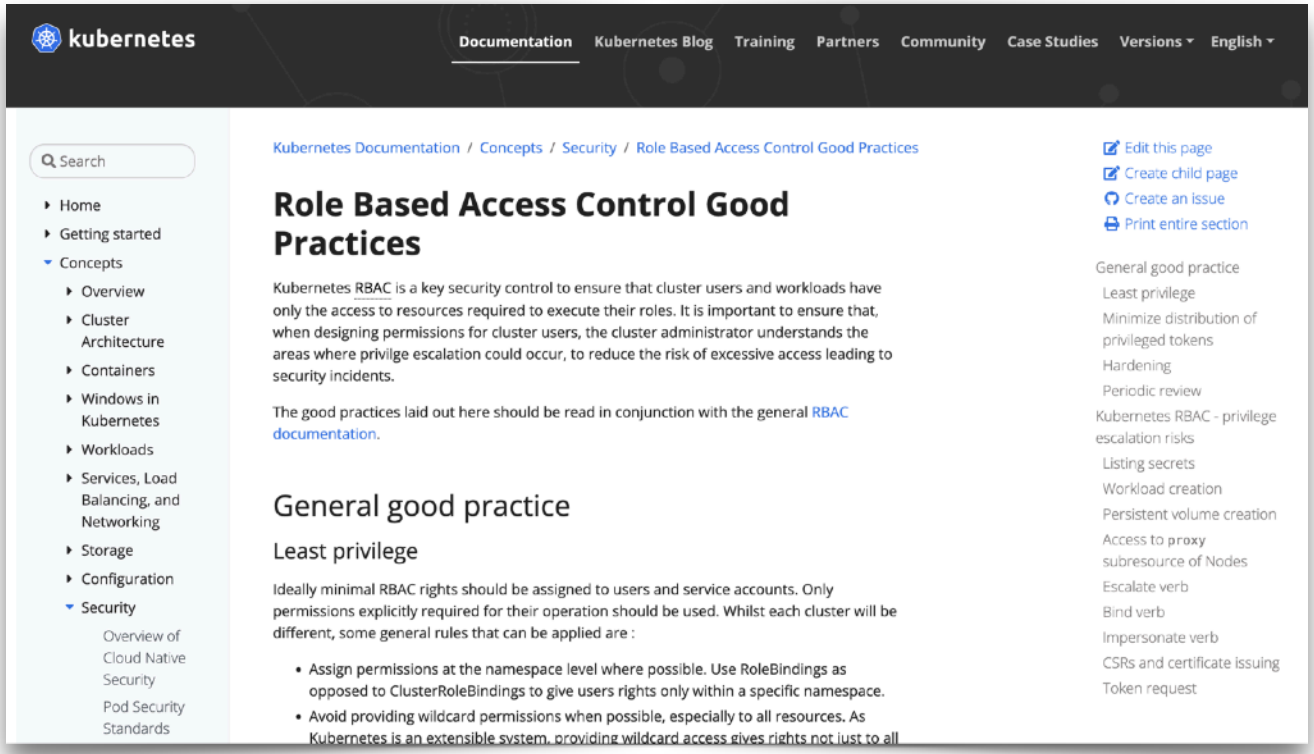
# Recent new releases

- Added new plugins: **cgroup**, **node**, **apiresources**,
- Added new flag checks from */proc/self/status*
- New support for **Linux aarch64** (and Darwin x86_64)
- You can now install kdigger with **Nix**!

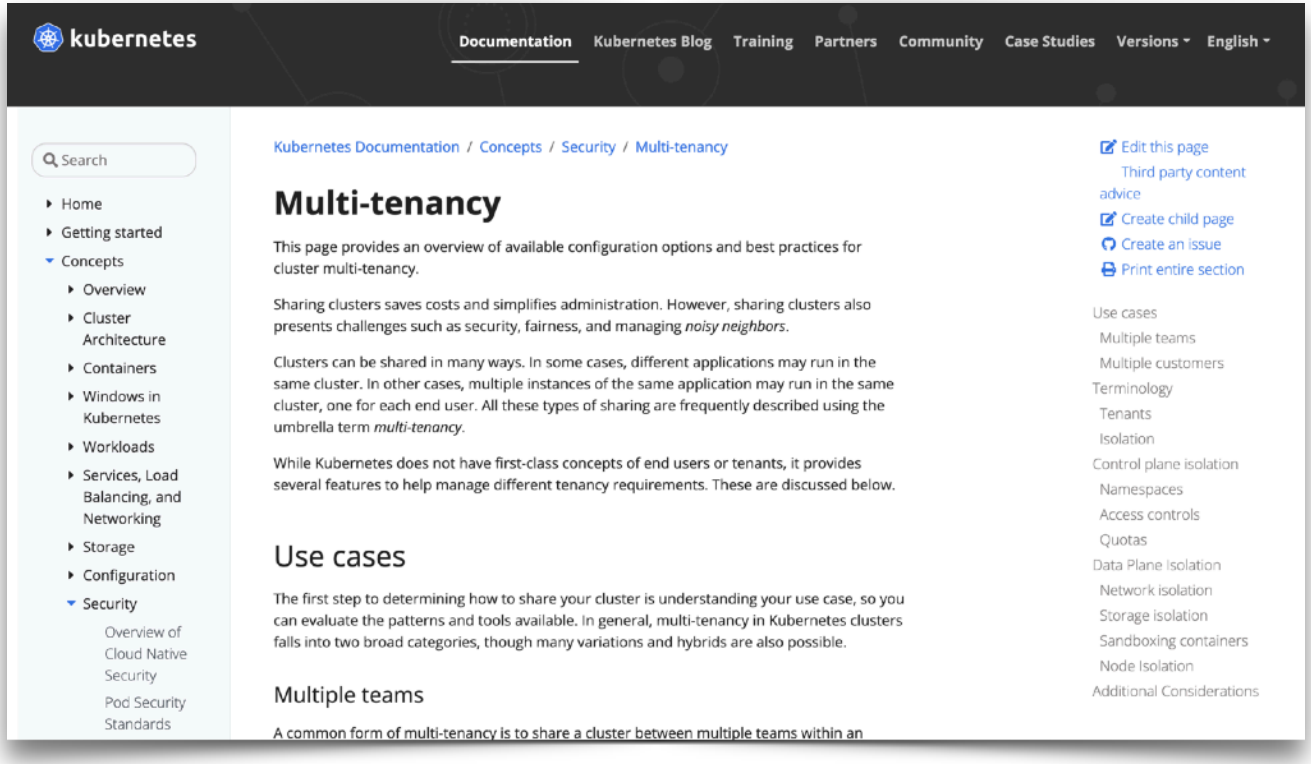
**Kubernetes community**
is working to improve
its **security**

Quarkslab

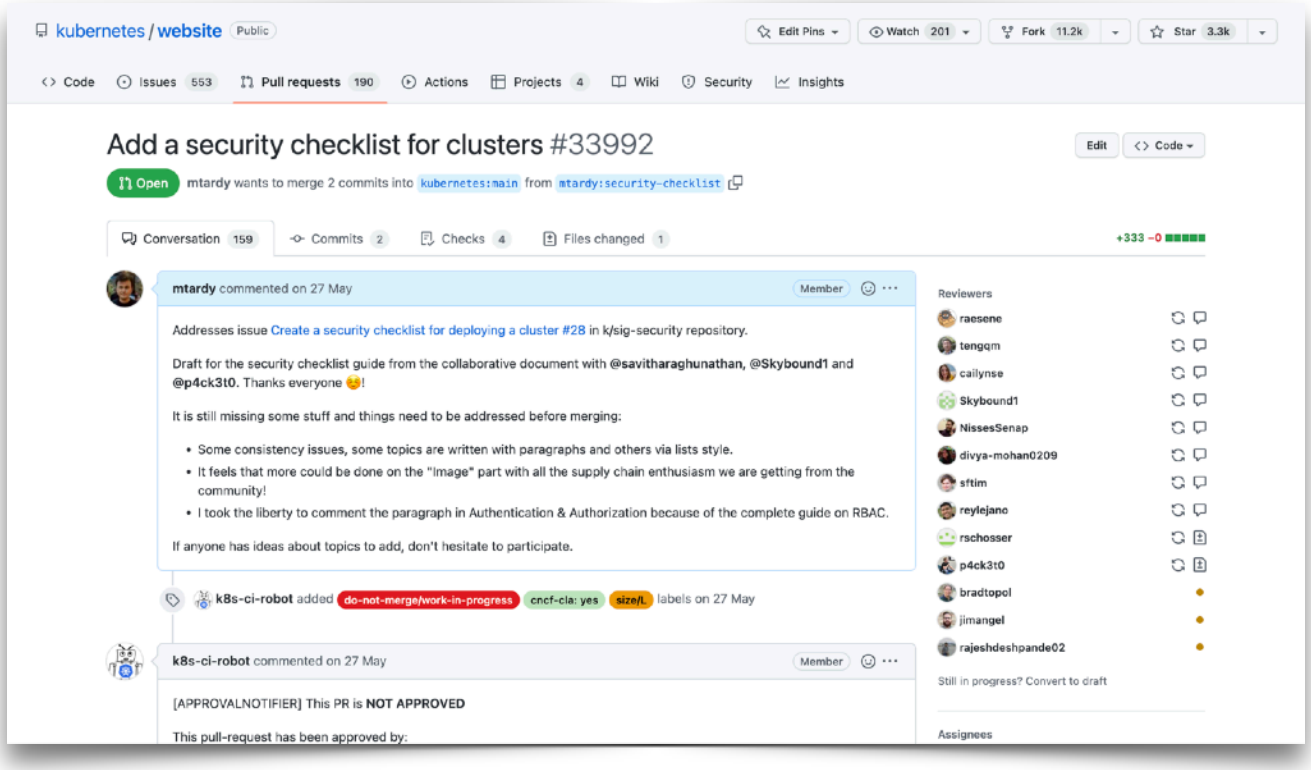
# New documentation on RBAC good practices (Mai 15)



# New documentation on multi-tenancy (June 23)



# New documentation on security checklist



# Thank you!

- Kubernetes focused **container** assessment and **context discovery** tool for **penetration testing**.
- **kdigger** is open-source and available at https://github.com/quarkslab/kdigger.
- **minik8s-ctf** is another open-source CTF project to directly experiment with **kdigger** https://github.com/quarkslab/minik8s-ctf.

Quarkslab

@mtardy_