A GLIMPSE INTO KUBERNETES SECURITY

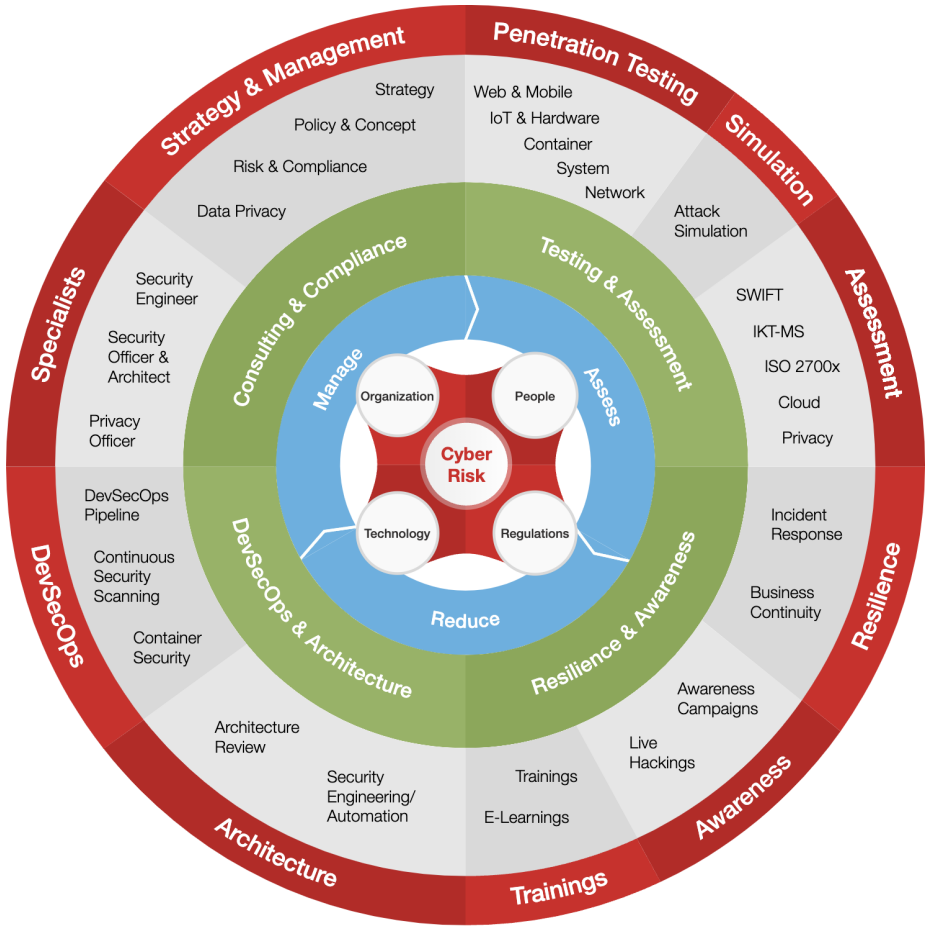
What could possibly go wrong?

$ whoami

- Sven Vetsch
- Redguard AG
  - Co-founder
  - Head of Innovation & Development
- Working in information security for 15+ years
- Specialized in application security (including containers, k8s, ...)

Contact: sven.vetsch@redguard.ch

Redguard



We also offer a full 2-day hands-on Kubernetes Security Training

&

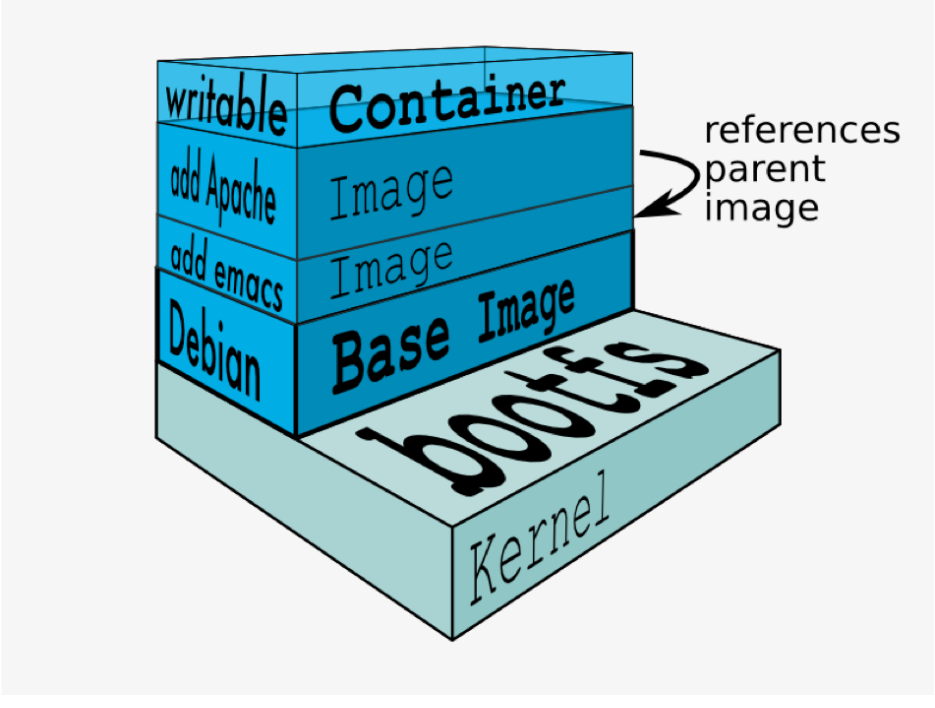
> By 2023, more than 70% of global organizations will be running more than two containerized applications in production, up from less than 20% in 2019
>
> http://www.gartner.com/document/3955920

CONTAINERS

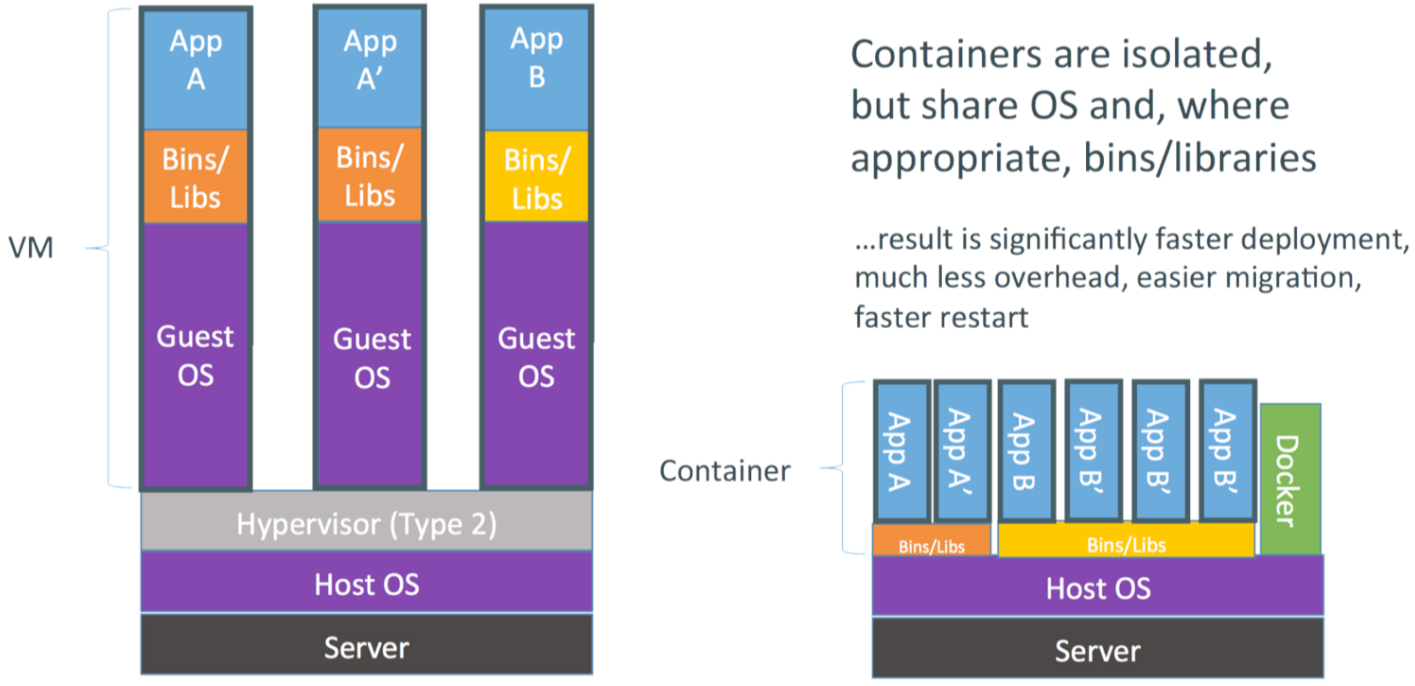
Just a quick recap

Images

Images are the underlying building blocks of each container.



Containers vs. Virtual Machines



Containers

- Containers are **not** Virtual Machines (VMs)
- Containers leverage abstraction of the OS (VMs leverage abstraction of hardware)
- Containers are “run”, VMs have to boot
- Container == runnable instance of an image
- Lightweight (kernel-based namespacing)
- Read-write (Copy-on-write (COW)) layer on top of the image

Containers

REDGUARD
SECURING YOUR ASSETS

Outside:

```
$ ps auxh | wc -l
177
```

Inside:

```
# ps aux
PID   USER     TIME   COMMAND
  1   root     0:00   sh
 11   root     0:00   ps aux
```

KUBERNETES

What are we even talking about?

What is k8s?

> Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.

https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

To make it easy ...

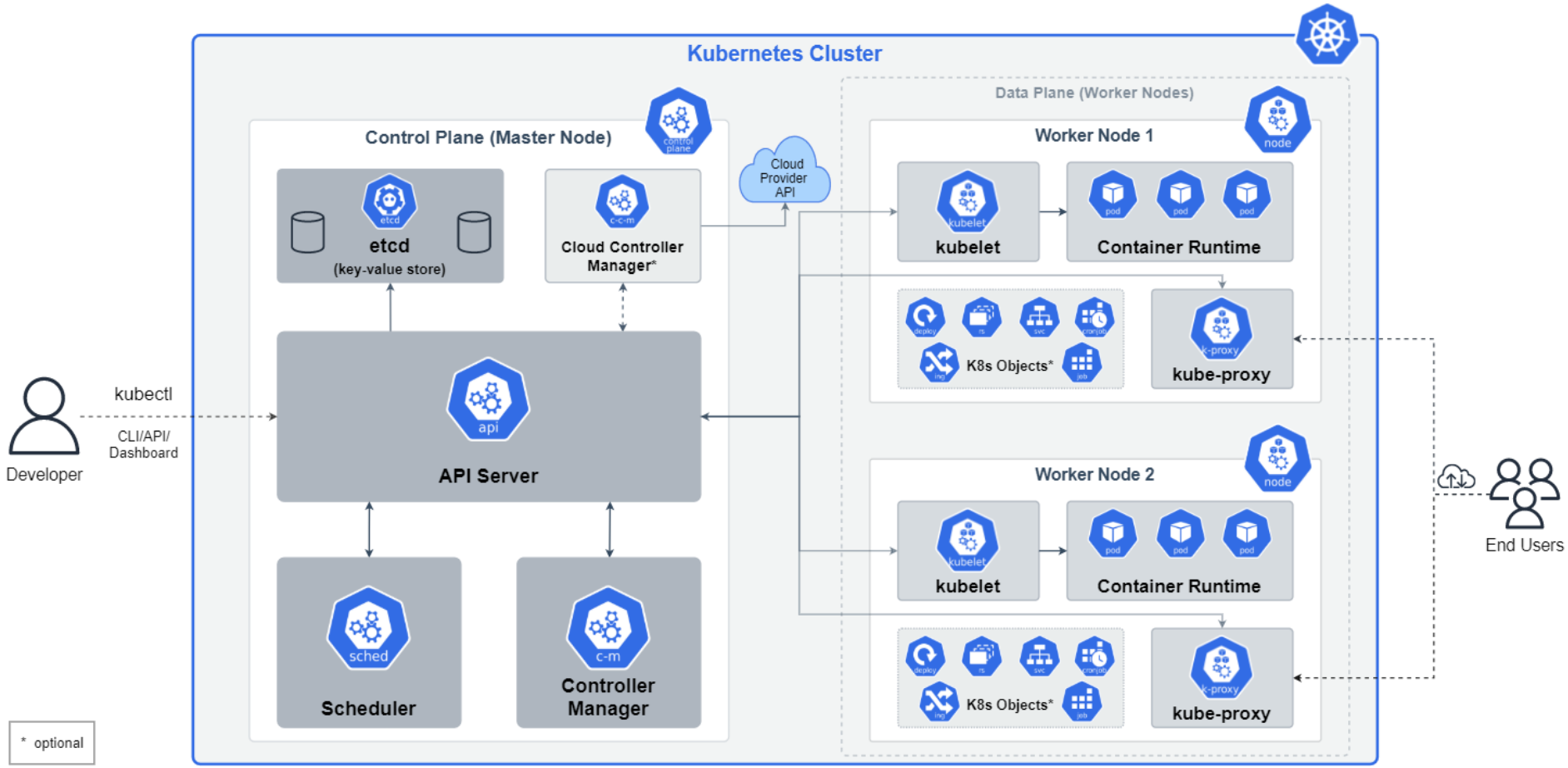
REDGUARD
SECURING YOUR ASSETS

Kubernetes

≅

General Purpose
Platform-as-a-Service (PaaS)
for containers

Kubernetes Archtitecture



ATTACK SURFACE

MITRE ATT&CK Threat Matrix

REDGUARD
SECURING YOUR ASSETS

A mapping to Kubernetes was done by Microsoft:

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evation	Credential Access	Discovery	Lateral Movement	Collection	Impact
Using Cloud Credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8s secrets	Access the K8s API server	Access cloud resources	Images from a private registry	Data destruction
Compromised images in registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8s events	Mount service principal	Access Kubelet API	Container service account		Resource Hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking		Denial of service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resource	Connect from proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files		
Exposed sensitive interfaces	Sidecar injection				Access managed identity credential	Instance metadata API	Writable volume mounts on the host		
					Malicious admission controller		CoreDNS poisoning		
							ARP poisoning and IP spoofing		

Blog: Secure containerized environments with updated threat matrix for Kubernetes

Redguard Kubernetes Threat Matrix

REDGUARD
SECURING YOUR ASSETS

Kubernetes Threat Matrix

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Impact
Using Cloud	Exec into Container	Backdoor Container	Privileged Container	Clear Container Logs	List K8s secrets	Access the K8s API server	Access cloud resources	Images from a private repository	Data Destruction
Compromised images in registry	bash/cmd in container	Writable hostPath mount	Cluster-admin binding	Delete k8s events	Mount service principal	Access Kubelet API	Container service account		Ressource hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking		Denial of Service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resources	Connect from proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files		
Exposed sensitive interfaces	SSH server running in inside container		Disable Namespacing		Access managed identity credentials	Instance metadata API	Writable volume mounts on the host		
	Sidecar injection				Malicious admission controller		CoreDNS poisoning		
							ARP poisoning and IP spoofing		

https://kubernetes-threat-matrix.redguard.ch/

Attack Demo

REDGUARD
SECURING YOUR ASSETS

Our attack path / kill chain:

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Impact
Using Cloud	Exec into Container	Backdoor Container	Privileged Container	Clear Container Logs	List K8s secrets	Access the K8s API server	Access cloud resources	Images from a private repository	Data Destruction
Compromised images in registry	bash/cmd in container	Writable hostPath mount	Cluster-admin binding	Delete k8s events	Mount service principal	Access Kubelet API	Container service account		Ressource hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking		Denial of Service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resources	Connect from proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files		
Exposed sensitive interfaces	SSH server running in inside container		Disable Namespacing		Access managed identity credentials	Instance metadata API	Writable volume mounts on the host		
	Sidecar injection				Malicious admission controller		CoreDNS poisoning		
							ARP poisoning and IP spoofing		

DEV(SEC)OPS CULTURE

It's not just about tech

The Organization

REDGUARD
SECURING YOUR ASSETS

- The most common "issue"
- Just by calling something Dev(Sec)Ops it doesn't magically become Dev(Sec)Ops
- Many companies currently only have 1-2 people who can handle Kubernetes
  - It's the same people that also have to take care of CI/CD pipelines and all the tooling
  - They also are the ones planning the architecture and setting up the systems
  - No quality control and if this single person leaves the company ... have fun
- DevSecOps doesn't mean that you had three people (Dev, Sec & Ops) and now one person can do all of that on his/her own. It's about bringing people closer together and getting rid of unnecessary boundaries.
- Responsibilities aren't clear (e.g. who defines what an acceptable firewall rule is?)
- We'll leave this here for now as we want to talk about Kubernetes but keep in mind that most of the time the current state of the organization is a key problem.

MAINTENANCE

Maintenance

- New (minor) version every four months (was three months before)
  - Too fast for many organizations.
  - This doesn't yet include all (security) patch releases
- Not enough experience nor automation to risk it.
- Updates must be done both on Kubernetes and the underlying nodes (idealy just replace them).
- Consider having *[Insert Favorite Cloud Provider]* maintaining your Kubernetes cluster(s)

HANDLING OF CONTAINERS & IMAGES

Handling of Containers & Images

REDGUARD
SECURING YOUR ASSETS

- It's the foundation. If the containers (including the applications running inside them) are insecure your Kubernetes cluster is at risk too.
- Hardening (examples):
  - Applications are secure on their own and packages up-to-date
  - No `privileged: true`
  - `automountServiceAccountToken: false`
  - Running as non-root user
  - Read-only root filesystem (not even once seen rolled out widely in production)
- No *Cloud Native (Security) Guidelines* at companies
- Vulnerability scanning of container images
  - One of the main things security teams want to have but most of them don't monitor the status once it is in place and they also don't enforce any restrictions besides on paper.

Container Image Scanning Example

REDGUARD
SECURING YOUR ASSETS

- Sooner or later vulnerabilities will be identified in packages that you've got in your images.
- Automated vulnerability scanning is the key to keep a clear view on the current security status.

```
$ trivy -s "LOW,MEDIUM,HIGH,CRITICAL" --ignore-unfixed redguard/lab

lab (debian 10.8)
=================================
Total: 37 (LOW: 4, MEDIUM: 6, HIGH: 22, CRITICAL: 5)
...

opt/bitnami/tomcat/webapps_default/ROOT.war (jar)
=================================================
Total: 8 (LOW: 0, MEDIUM: 2, HIGH: 5, CRITICAL: 1)
...
```

Pod (Resource) Limits

REDGUARD
SECURING YOUR ASSETS

- Limiting CPU und memory consumption
- Makes sure that pods/containers can't cause a denial of service situation.

At least you should use:

- `AllowPrivilegeEscalation = false`
- `ReadOnlyRootFileSystem = true`
- `RunAsNonRoot = true`

If you'd like to go even further:

- Seccomp
- AppArmor
- SELinux

SEGREGATION

There's more than namespaces

Segregation

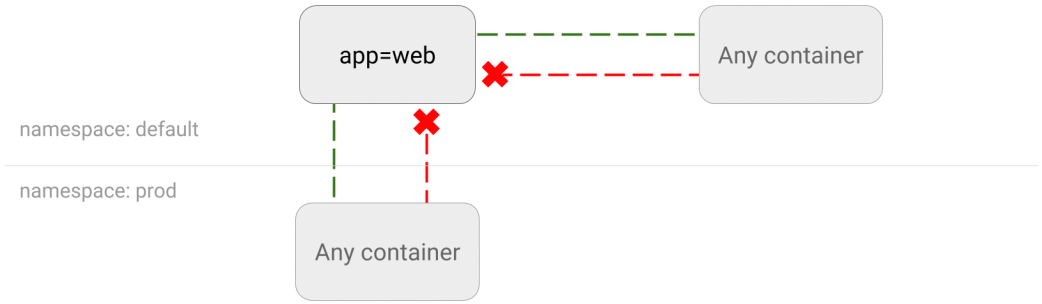
- Permissions / Accessibility (**RBAC**)
- Data / Storage
- CPU / Memory
- Network
  - Nearly everywhere we've seen that any pod/container can still reach the kube-api and e.g. etcd which shouldn't be the case.
- Namespaces
- Nodes (Taints / Tolerations)

Network Security

REDGUARD
SECURING YOUR ASSETS

- By default there's one flat network (every pod can reach everything)
  - This includes the kube-api, etcd, ...
  - In "old" networks we e.g. also don't want a web-app frontend to be able to directly talk to a database.
- Use network policies.
- Container Network Interface (CNI) plugins like *Cilium* can provide a lot more granular policies than the normal *NetworkPolicy*
- A Service Mesh like *Istio* can add better visibility and at the same time increase security (e.g. mTLS between pods)
- Keep in mind that old network attacks like IP spoofing or ARP poisoning in general still work in Kubernetes.

NetworkPolicy Example



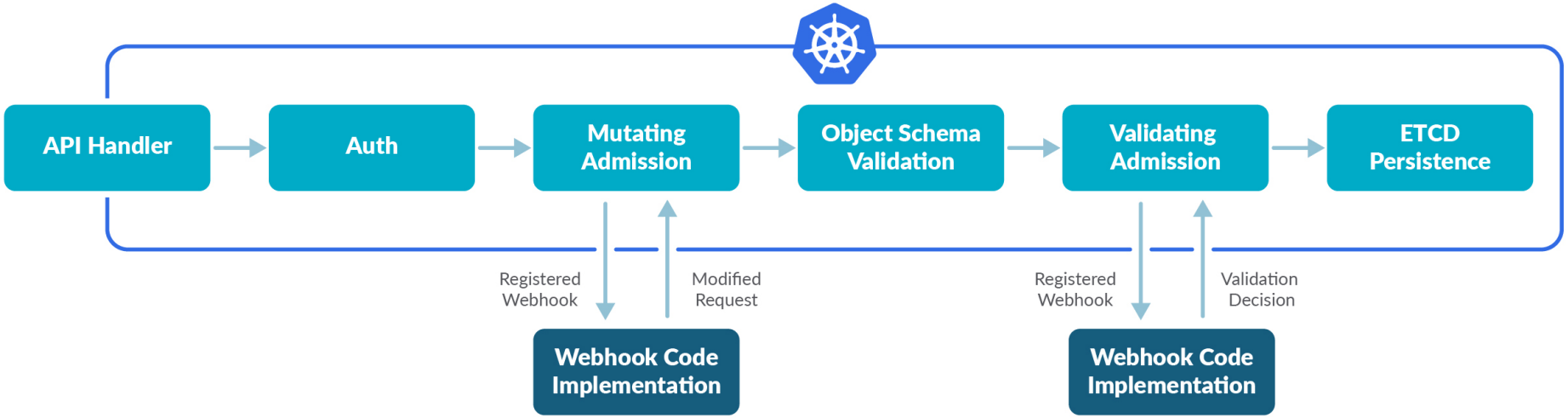

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-deny-all
spec:
  podSelector:
    matchLabels:
      app: web
  ingress: []
```

https://github.com/ahmetb/kubernetes-network-policy-recipes

ADMISSION CONTROLLER

Policy as Code in Kubernetes

Admission Controller



- Basically one of the main security layer of Kubernetes
- Most used/known admission controller was *PodSecurityPolicy* (PSP)
  - Removed in Kubernetes v1.25
  - Replaced by *Pod Security Admission* (PSA) / Pod Security Standards (PSS)

PodSecurityPolicy ⚠️ (not available anymore)

REDGUARD
SECURING YOUR ASSETS

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: restricted
spec:
  readOnlyRootFilesystem: true
  privileged: false
  allowPrivilegeEscalation: false
  runAsUser:
    rule: 'MustRunAsNonRoot'
  supplementalGroups:
    rule: 'MustRunAs'
    ranges:
      - min: 1
        max: 65535
  fsGroup:
    rule: 'MustRunAs'
    ranges:
      - min: 1
        max: 65535
  seLinux:
    rule: 'RunAsAny'
  volumes:
  - configMap
  - emptyDir
  - secret
```

Pod Security Standards

REDGUARD
SECURING YOUR ASSETS

There are three different profiles for the Pod Security Standards:

Privileged: Unrestricted policy, providing the widest possible level of permissions. (DANGER ZONE)

Baseline: Minimally restrictive policy which prevents known privilege escalations.

Restricted: Heavily restricted policy, following current Pod hardening best practices.

```
$ kubectl create ns pss-demo
namespace/pss-demo created
$ kubectl label ns pss-demo 'pod-security.kubernetes.io/enforce=baseline'
namespace/pss-demo labeled
$ kubectl get ns pss-demo --show-labels
NAME       STATUS   AGE    LABELS
pss-demo   Active   2m9s   ...,pod-security.kubernetes.io/enforce=baseline
```

Pod Security Standards

REDGUARD
SECURING YOUR ASSETS

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: privileged-pod
  namespace: pss-demo
spec:
  containers:
  - name: privileged
    image: nginx
    securityContext:
      privileged: true
EOF
```

Pod Security Standards

REDGUARD
SECURING YOUR ASSETS

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: privileged-pod
  namespace: pss-demo
spec:
  containers:
  - name: privileged
    image: nginx
    securityContext:
      privileged: true
EOF
```

```
... pods "privileged-pod" is forbidden: violates PodSecurity "baseline:latest": privileged
  (container "privileged" must not set securityContext.privileged=true)
```

Open Policy Agent (OPA)

REDGUARD
SECURING YOUR ASSETS

- OPA Gatekeeper - Policy Controller for Kubernetes

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: all-must-have-owner
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Namespace"]
  parameters:
    message: "All namespaces must have an `owner` label"
    labels:
      - key: owner
        allowedRegex: "^[a-zA-Z]+.example.demo$"
```

Open Policy Agent (OPA)

REDGUARD
SECURING YOUR ASSETS

Example in plain Rego:

```
package kubernetes.admission

deny[msg] {
    input.request.kind.kind == "Pod"
    some i
    image := input.request.object.spec.containers[i].image
    not startswith(image, "example.com/")
    msg := sprintf("image '%v' comes from untrusted registry", [image])
}
```

Open Policy Agent (OPA)

REDGUARD
SECURING YOUR ASSETS

Example in plain Rego:

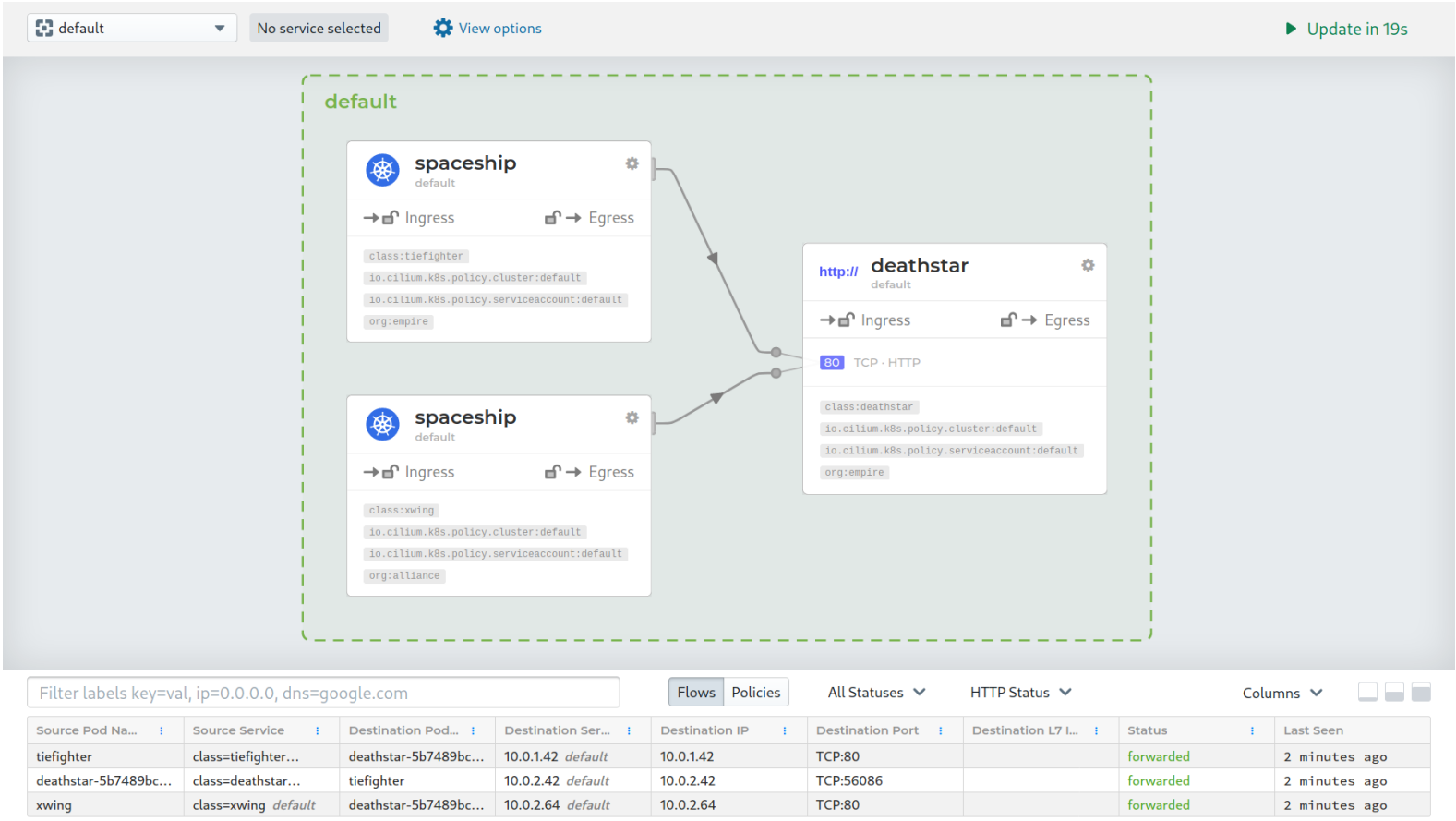
```
package kubernetes.admission

deny[msg] {
    input.request.kind.kind == "Pod"
    some i
    image := input.request.object.spec.containers[i].image
    not startswith(image, "example.com/")
    msg := sprintf("image '%v' comes from untrusted registry", [image])
}
```

There are of course alternatives to OPA (like e.g. *Kyverno*).

MONITORING AND VISIBILITY

Know what's going on

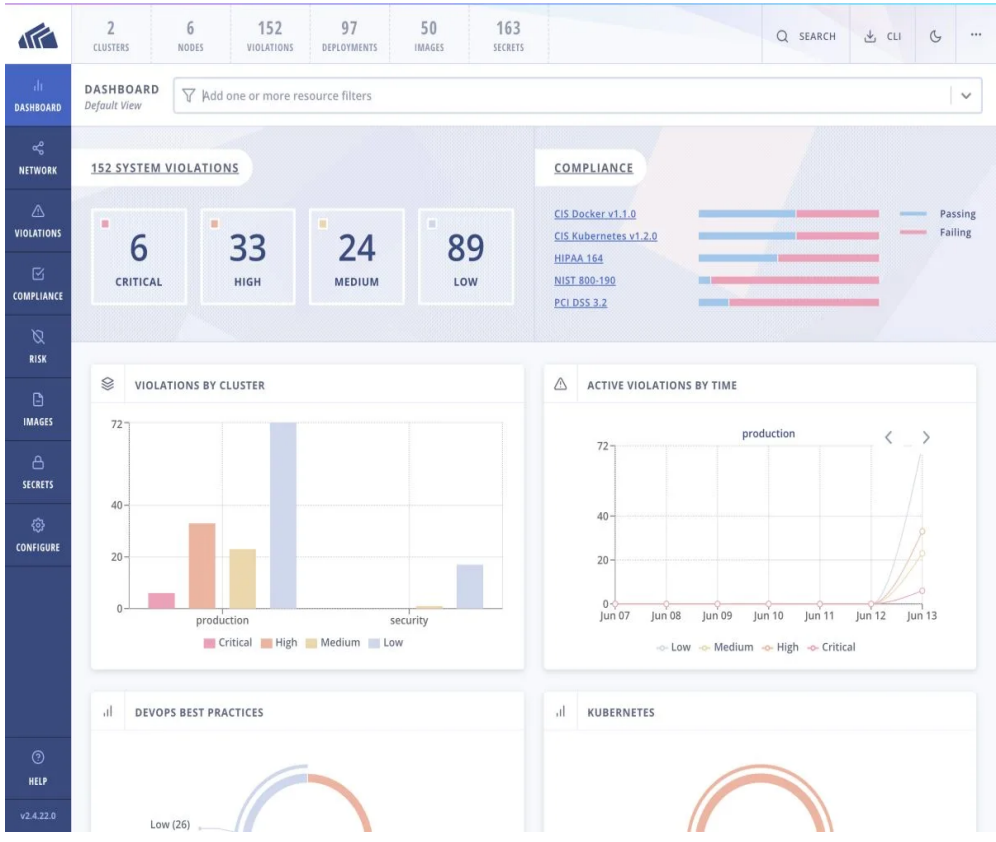
Network (Policies)



Cilium Service Map & Hubble UI

Cluster / Pods / Containers

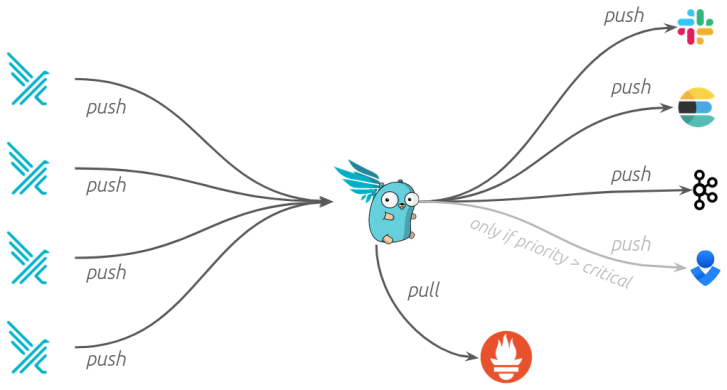
- Prometheus / Grafana
- Elasticsearch / Fluentd / Kibana (EFK)
- Splunk
- StackRox
- Portshift
- dynatrace
- OpenShift
- Sysdig
- Starboard
- Octant
- Lens
- ...



Falco

- Monitoring all the way down to Linux syscalls if you like.
- Simple anomaly detection due to containers.
- Generates alerts when a treat is detected
- Simple but powerfull rule engine

IMHO the one must-have security monitoring tool for Kubernetes.
(Even more so in combination with *Falcosidekick*)



Audit Log

- The component to log (security relevant) events in Kubernetes (or more specific the Kubernetes API)
- Can basically catch everything
- Not enabled by default
- Needs tuning to get the "right amount" of information
  - Else you can expect Gigabytes of audit logs per day
- Is very valuable in case of an incident.
  - Or even just to do normal debugging of the cluster.

Summary

- Have a great team that knows what they're doing (they likely need proper training)
- Perform basic hardening on containers, nodes and Kubernetes on a regular basis
- Use RBAC with an appropriate permission model
- Have a guideline/policy in place that explains the required security measures and what their actual purpose is
- Use admission controllers to enforce your security requirements
- Have a working security monitoring in place

Take Away

REDGUARD
SECURING YOUR ASSETS

Kuberetes scales
but so do it's security issues

Q & A

What more do you want to know about k8s security?

K8S HACKING CHALLENGE

Let's get to work