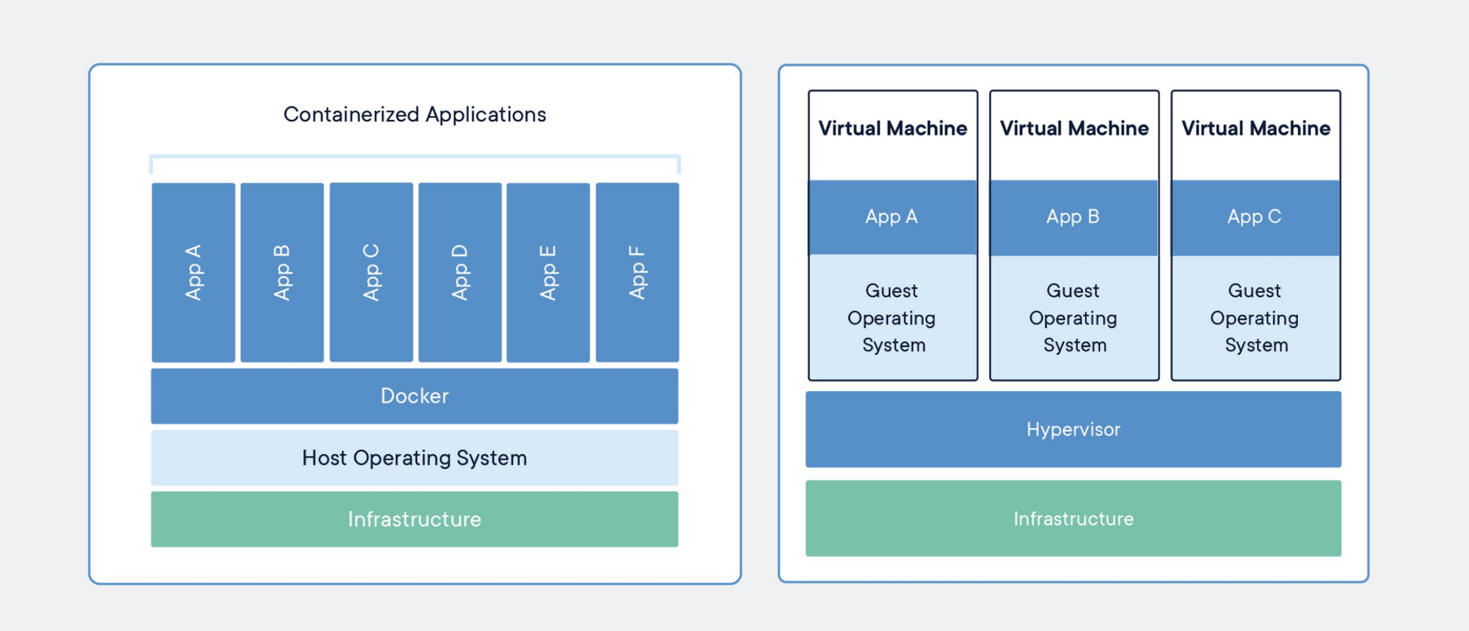
Certified Kubernetes Security Specialist

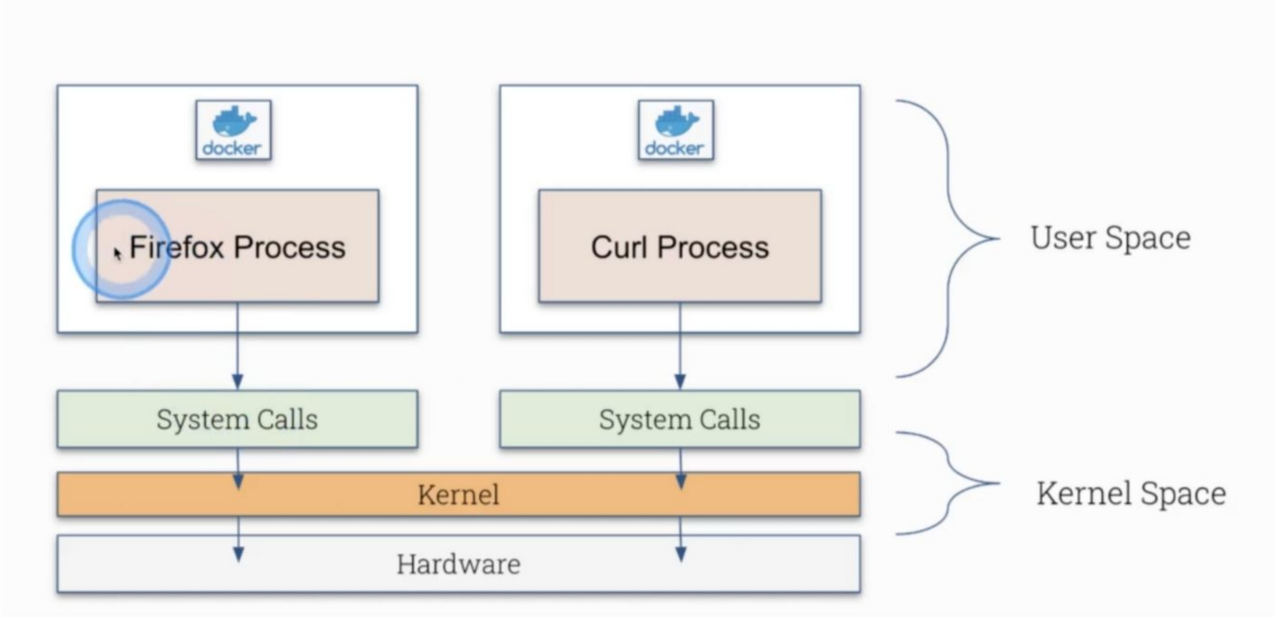
By Nilesh Jayanandana

Introduction

Containers vs VMs



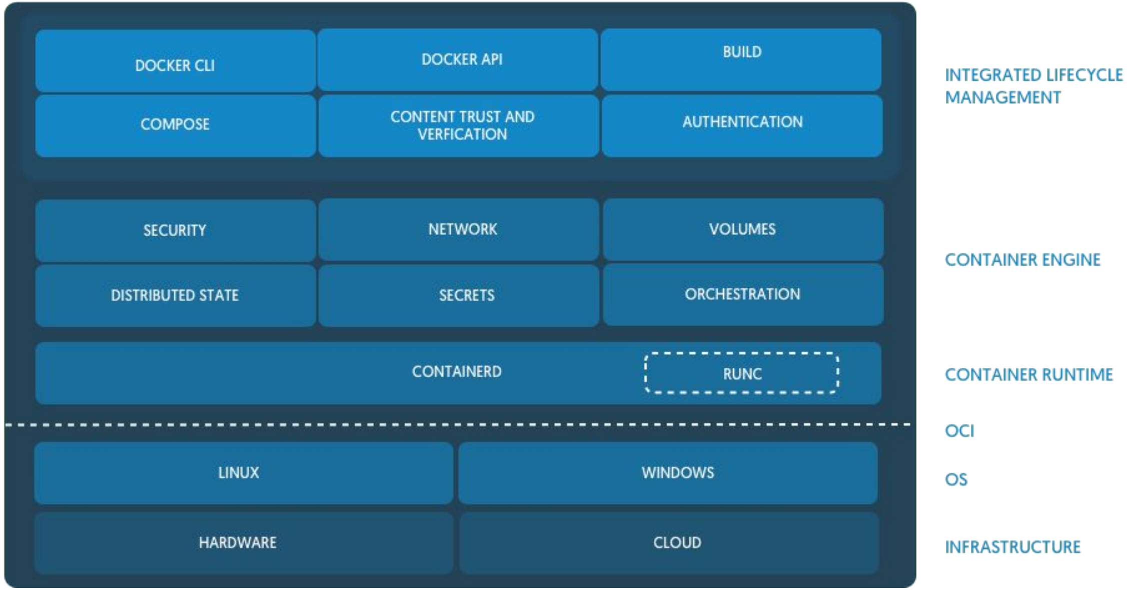
Kernel Space vs User Space



Container Runtimes

- Containerd
- Rkt
- CRI-O

Docker CE is made of Containerd + RunC



DOCKER CLI | DOCKER API | BUILD

COMPOSE | CONTENT TRUST AND VERFICATION | AUTHENTICATION

INTEGRATED LIFECYCLE MANAGEMENT

SECURITY | NETWORK | VOLUMES

DISTRIBUTED STATE | SECRETS | ORCHESTRATION

CONTAINER ENGINE

CONTAINERD | RUNC

CONTAINER RUNTIME

OCI

LINUX | WINDOWS

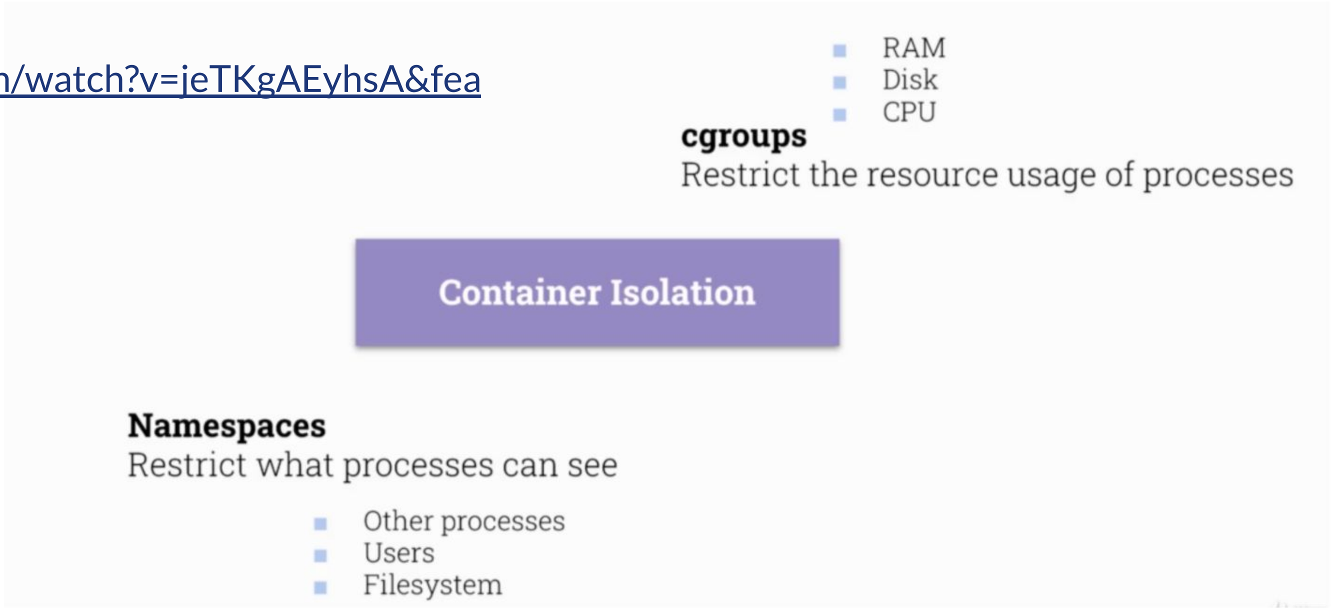
OS

HARDWARE | CLOUD

INFRASTRUCTURE

Container Isolation

Additional https://www.youtube.com/watch?v=jeTKgAEyhsA&feature=emb_title



- RAM
- Disk
- CPU

cgroups
Restrict the resource usage of processes

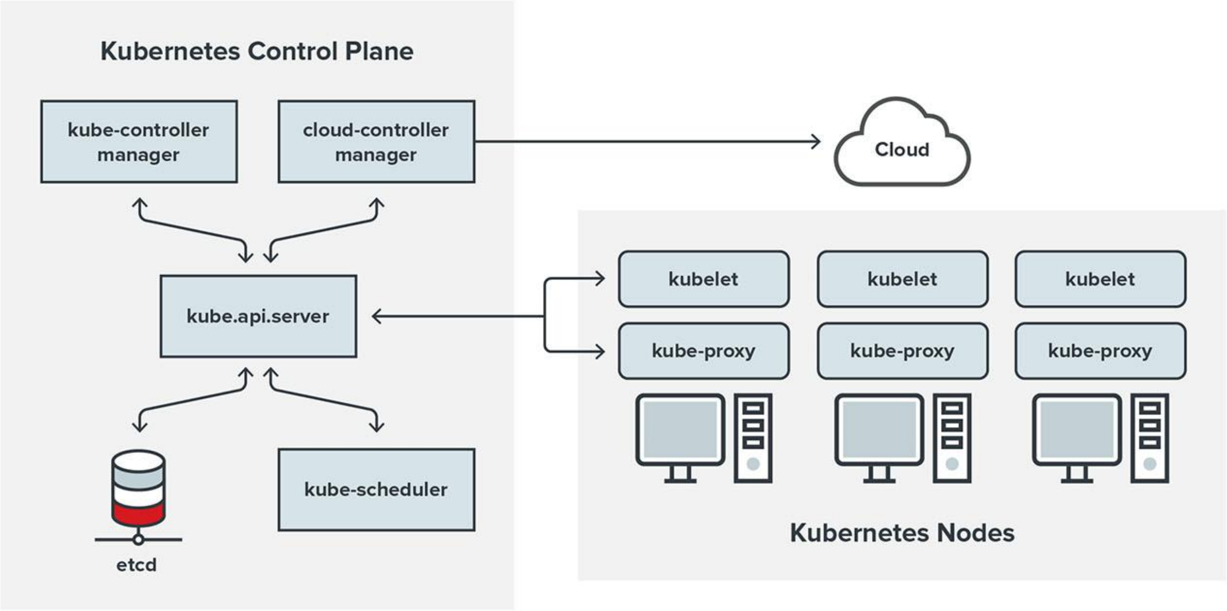
Container Isolation

Namespaces
Restrict what processes can see

- Other processes
- Users
- Filesystem

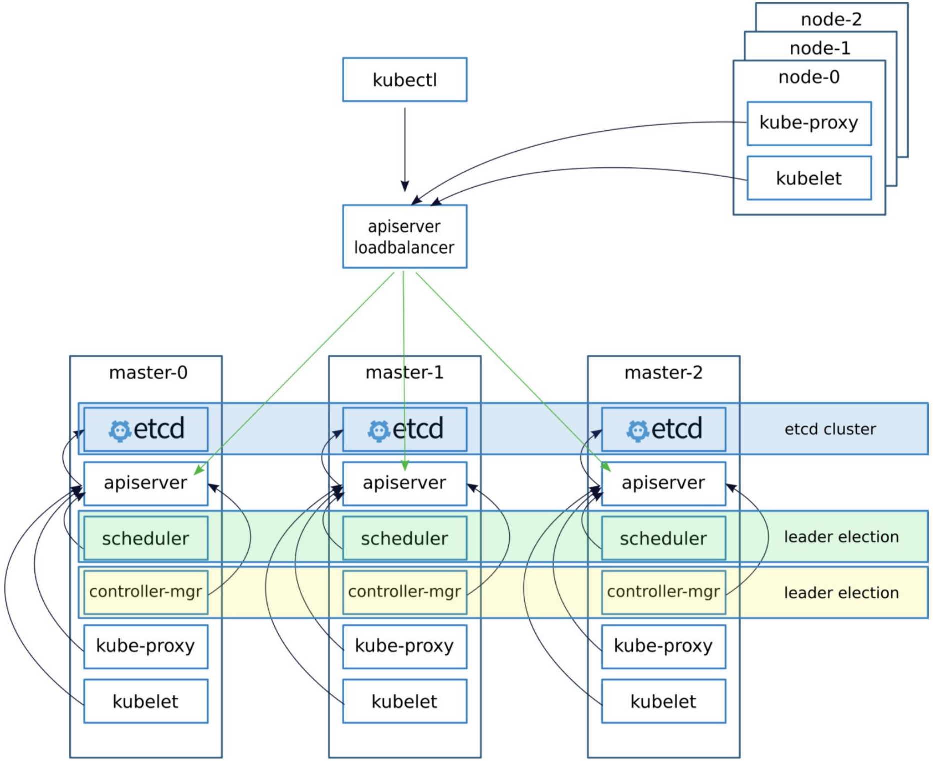
Kubernetes Components

- etcd
- Kubelet
- Scheduler
- Controller Manager
- Kube-DNS (Core dns)
- Kube-Proxy
- Kube API Server

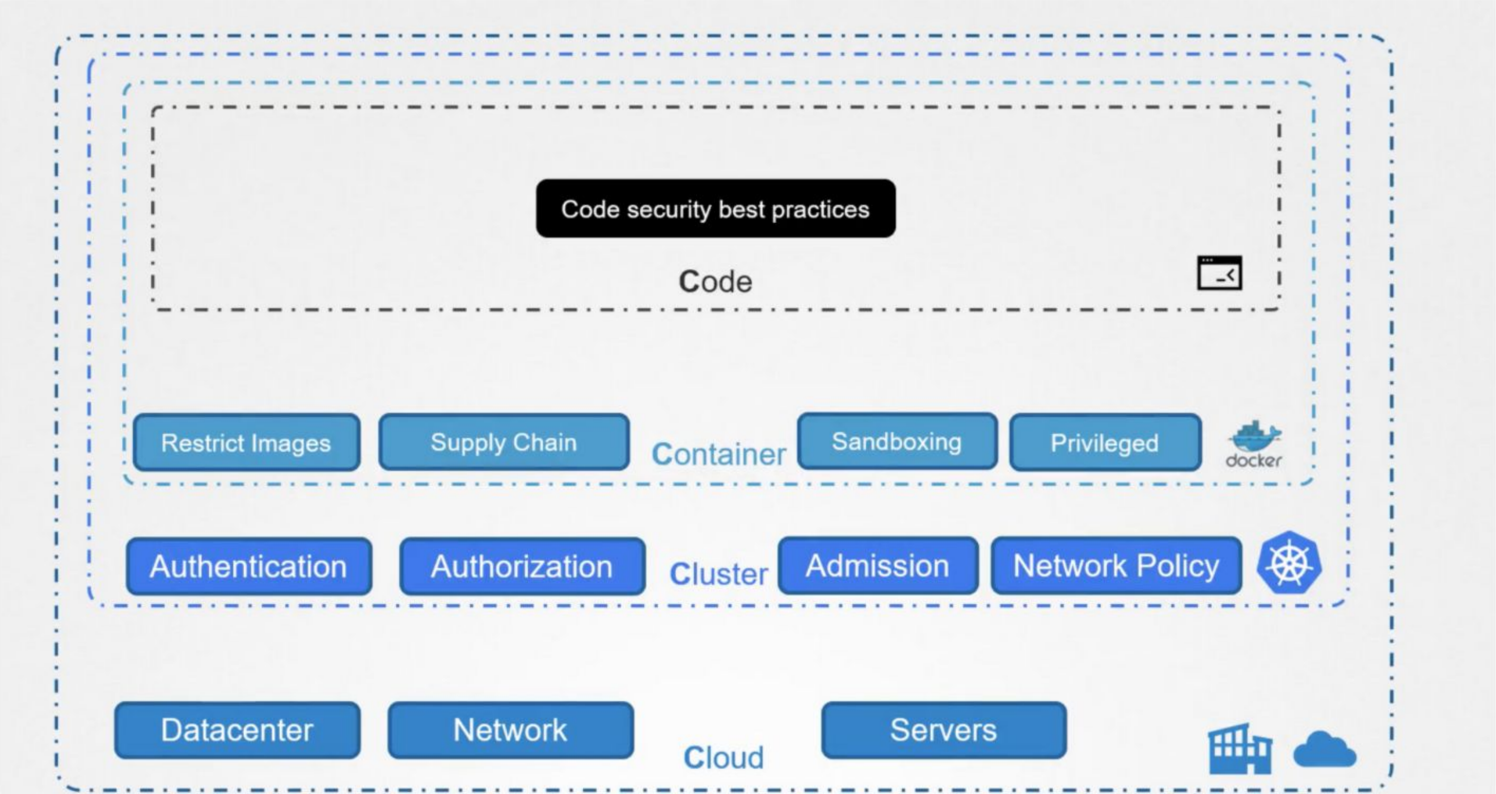


Kubernetes Deployment

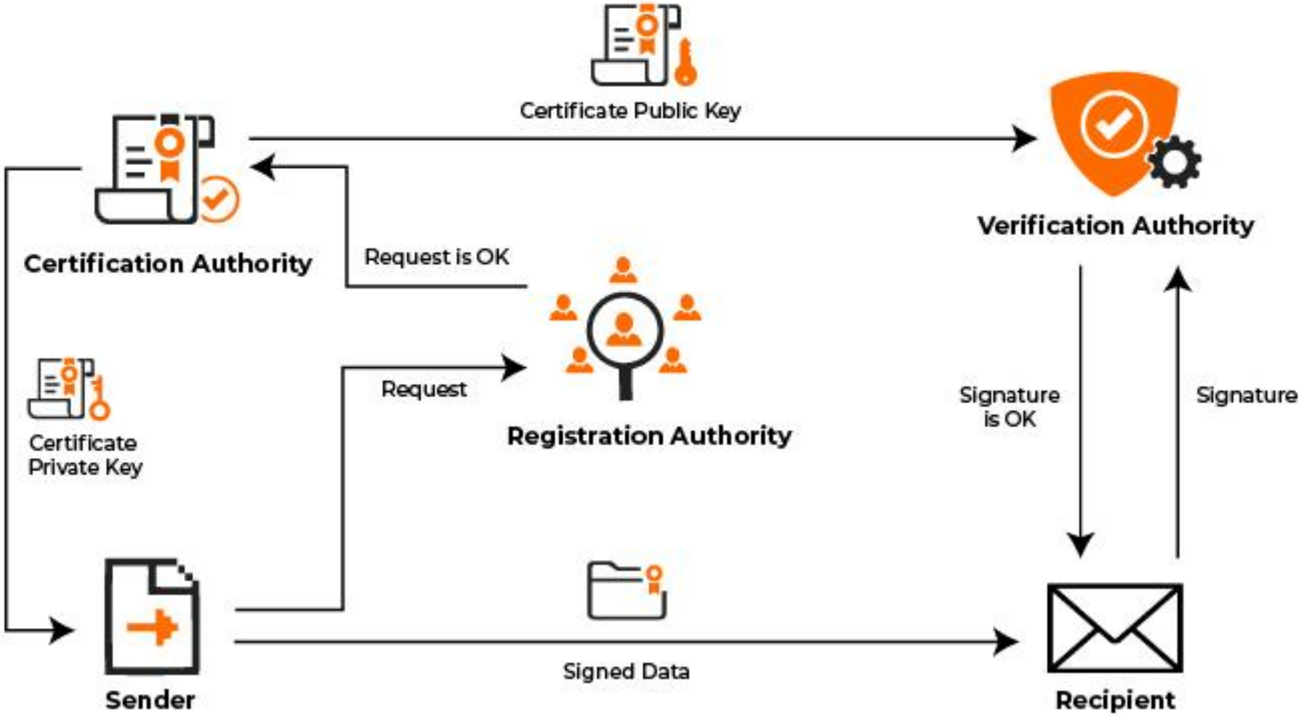
- Single Master Cluster
- HA with Stacked ETCD
- HA with external ETCD



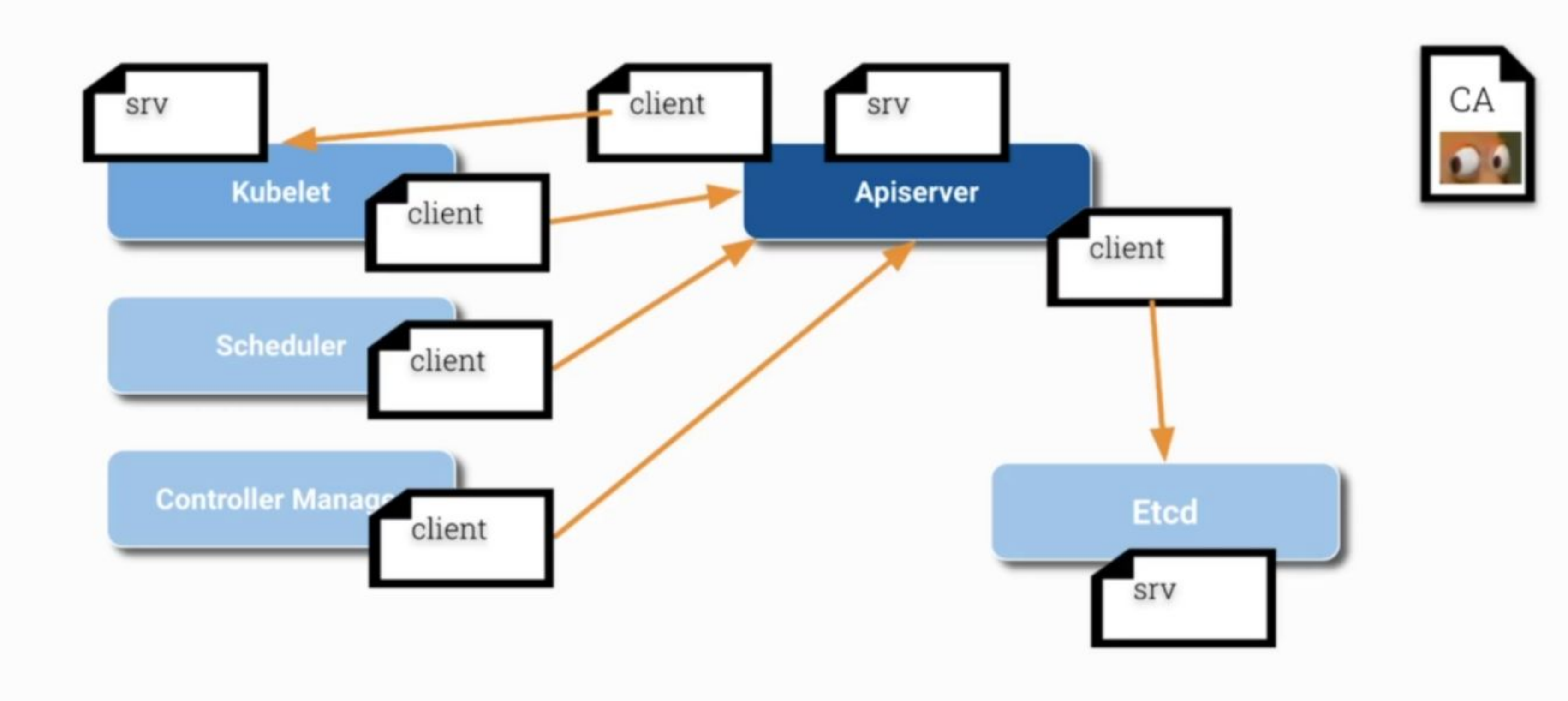
4Cs of Security Layers



Public Key Infrastructure



Public Key Infrastructure (PKI) for Kubernetes



Ports Needed to be Open in Kubernetes

- Master
  - 6443 - api server
  - 2379-2380 - etcd
  - 10250 - kubelet
  - 10251 - scheduler
  - 10252 - controller-manager
  - 10255 - kubelet read only
  - 8472 UDP - kube proxy
  - 30000-32767 - node ports

- Worker
  - 10250 - kubelet
  - 10255 - kubelet readonly
  - 8472 UDP - kube proxy
  - 30000-32767 - nodeports

Setup Kubernetes Cluster

1. Install ContainerD/Docker
2. Install Kubeadm, Kubectl
3. Open Ports
4. Initialize Kubernetes master with Kubeadm
5. Initialize Kubernetes nodes with Kubeadm
6. Install CNI

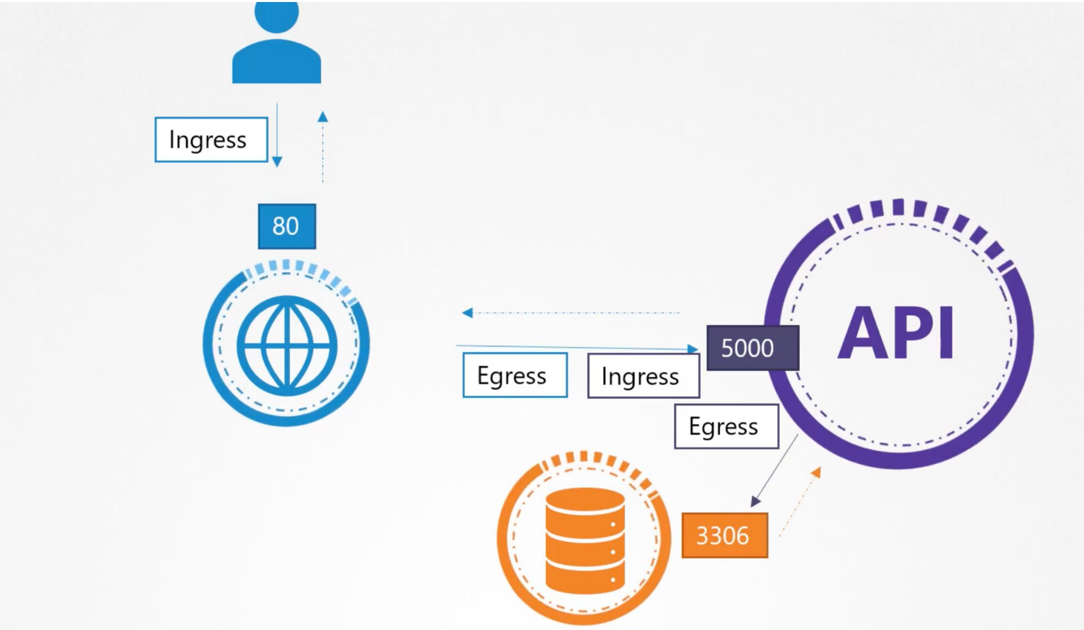
Install Script:
https://gist.githubusercontent.com/nilesh93/fe90c8d2137bc24d32479e4fae64c558/raw/9db75cbf4e211c23c9647dc41d1d29e45fc16f41/kubernetes-prerequisites-ubuntu.sh

CNI Reference: Additional Reading
https://www.slideshare.net/JurajHantak/4-cncf-kubernetes-comparison-ofexistingcnipluginsforkubernetes?from_action=save

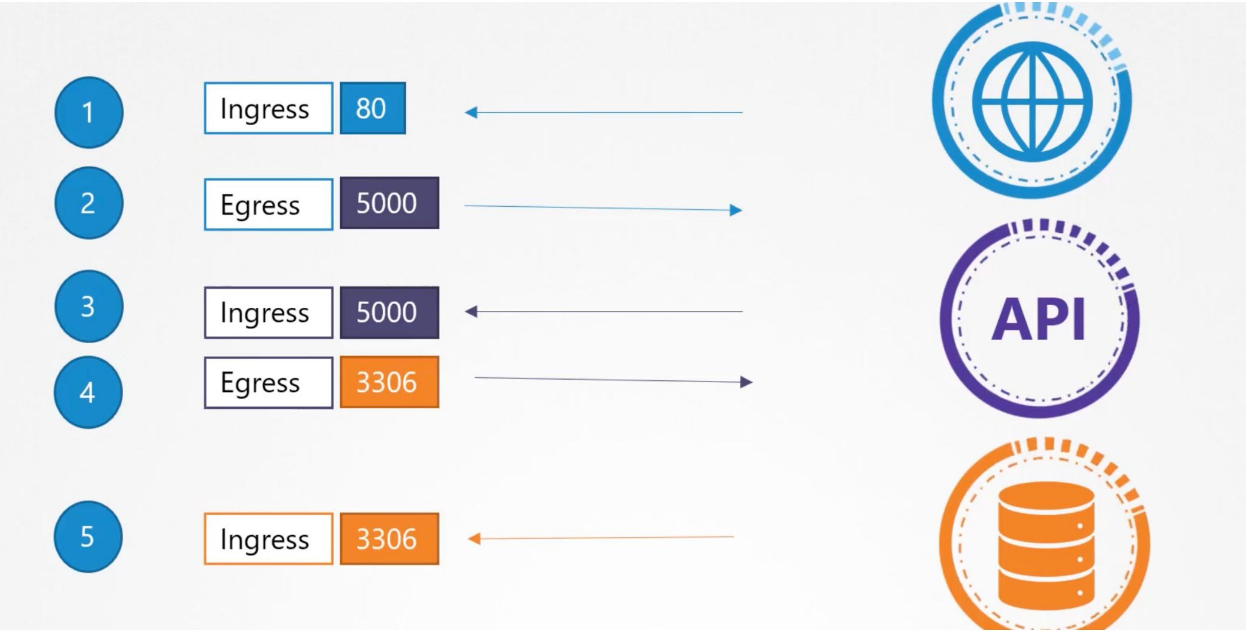
Network Policies

Network Policies

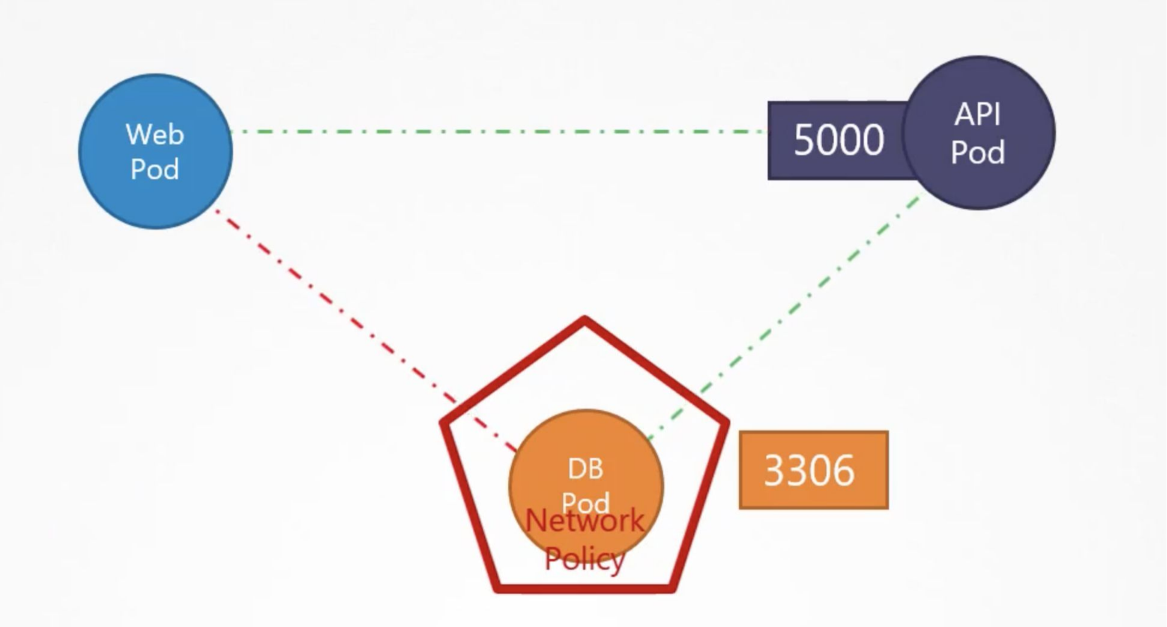
- Ingress - Traffic coming in to the pod
- Egress - Traffic going out of the pod
- Can limit via
  - Pod Selectors
  - Namespace selectors
  - IP Ranges

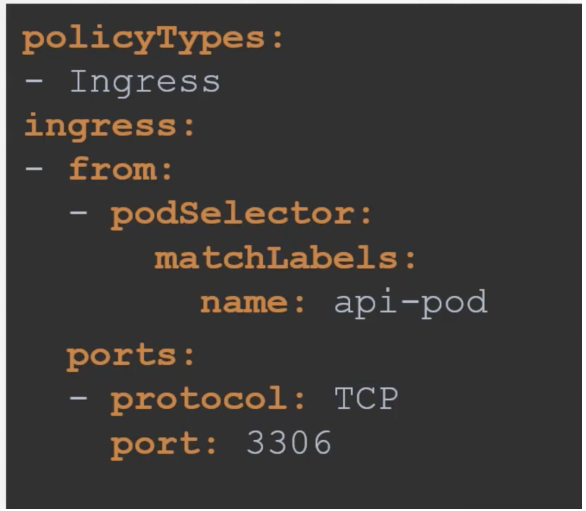


Traffic Rules



Traffic Rules with Network Policy

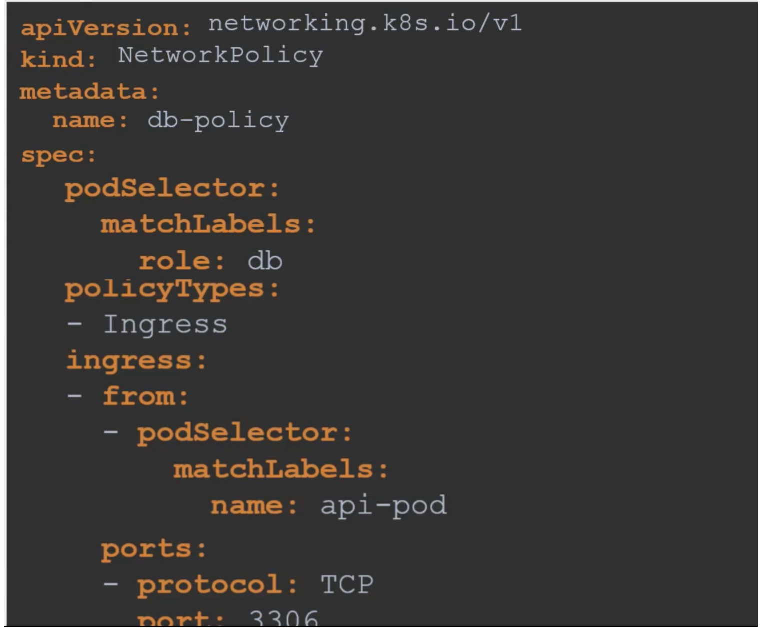




```
policyTypes:
- Ingress
ingress:
- from:
  - podSelector:
      matchLabels:
        name: api-pod
  ports:
  - protocol: TCP
    port: 3306
```

Network Policy Full Example

Additional Information:
https://kubernetes.io/docs/concepts/services-networking/network-policies/



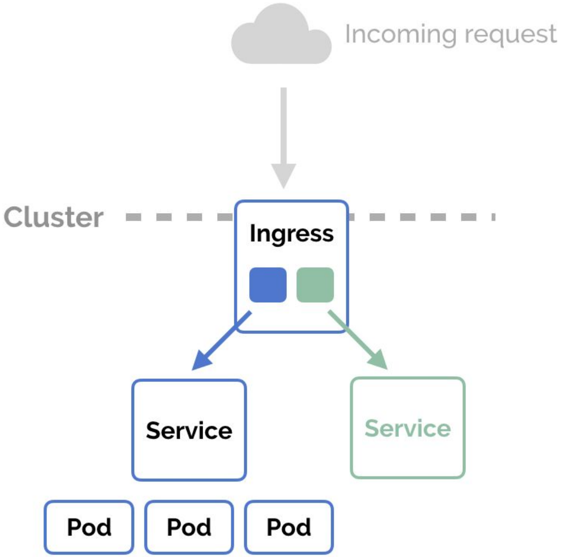
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: db-policy
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          name: api-pod
    ports:
    - protocol: TCP
      port: 3306
```

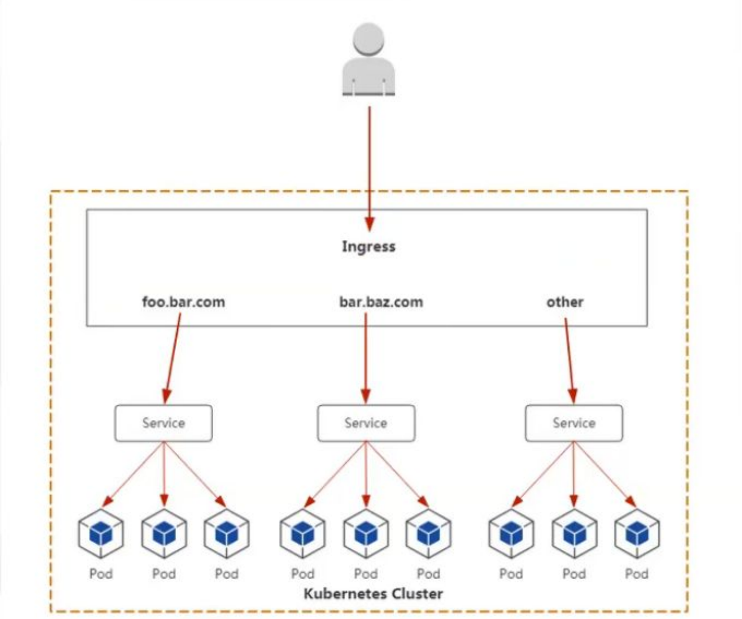
CNIs that Support Network Policies

- Calico
- Cillium
- Kube Router
- Canal
- Weavenet

****Flannel Does not support Network Policies**

Ingress Objects and SSL





CIS Benchmarking

CIS Benchmark

- We use Kubernetes Benchmark v1.16.pdf
- Check supported k8s versions in the first page
- Control Plane recommendations - page 16
- Node Recommendations - Page 208

Get the latest benchmark here
https://www.cisecurity.org/benchmark/kubernetes/

Kube Bench

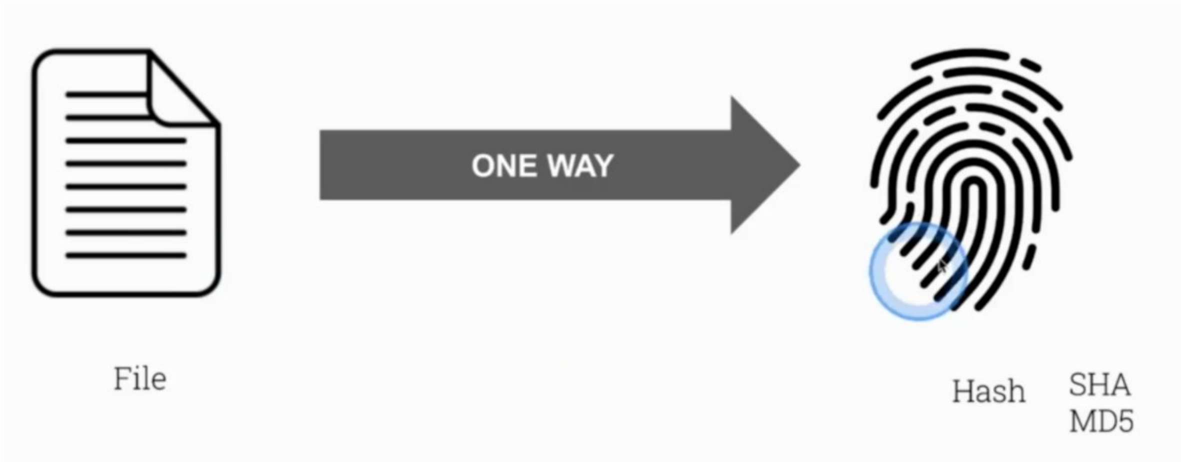
Developed by Aquasec, a tool to benchmark Kubernetes clusters and apply recommendations

https://github.com/aquasecurity/kube-bench

Docker Run Command

```
docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -t
aquasec/kube-bench:latest --version 1.18
```

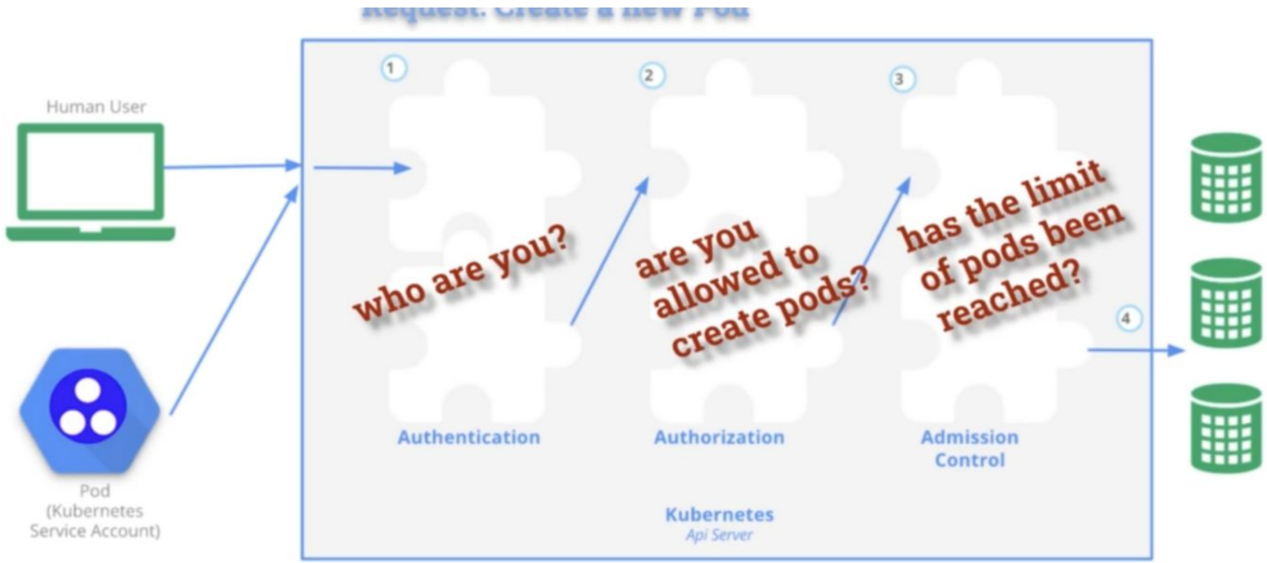
Binary Verification



CODE: *sha512sum <filename>*

Cluster Hardening

API Request Flow

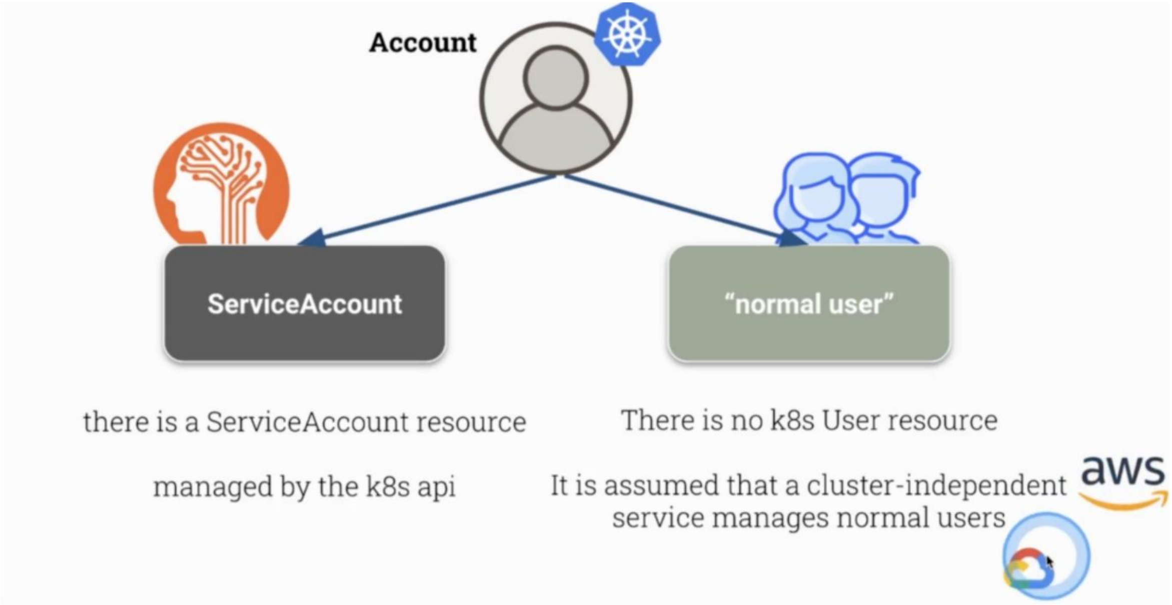


Restrictions

- Don’t allow anonymous access (--anonymous-auth=false)
- Close insecure ports (--insecure-port=0)
- Don’t expose API to outside
- Restrict access from nodes to API (--enable-admission-plugins=NodeRestriction)
- Prevent Unauthorized access (RBAC)
- Prevent Pods accessing API
- API server behind a firewall and ip whitelisted range in cloud

References: https://kubernetes.io/docs/concepts/security/controlling-access/

Kubernetes Users



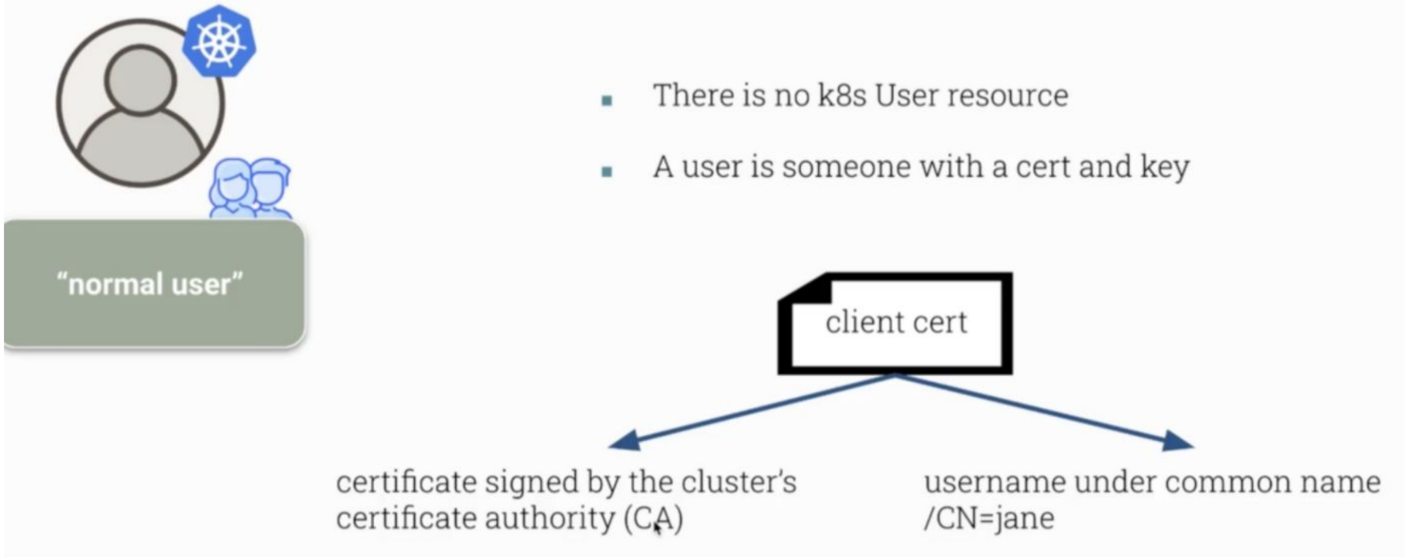
there is a ServiceAccount resource

managed by the k8s api

There is no k8s User resource

It is assumed that a cluster-independent service manages normal users

User Certificates



"normal user"

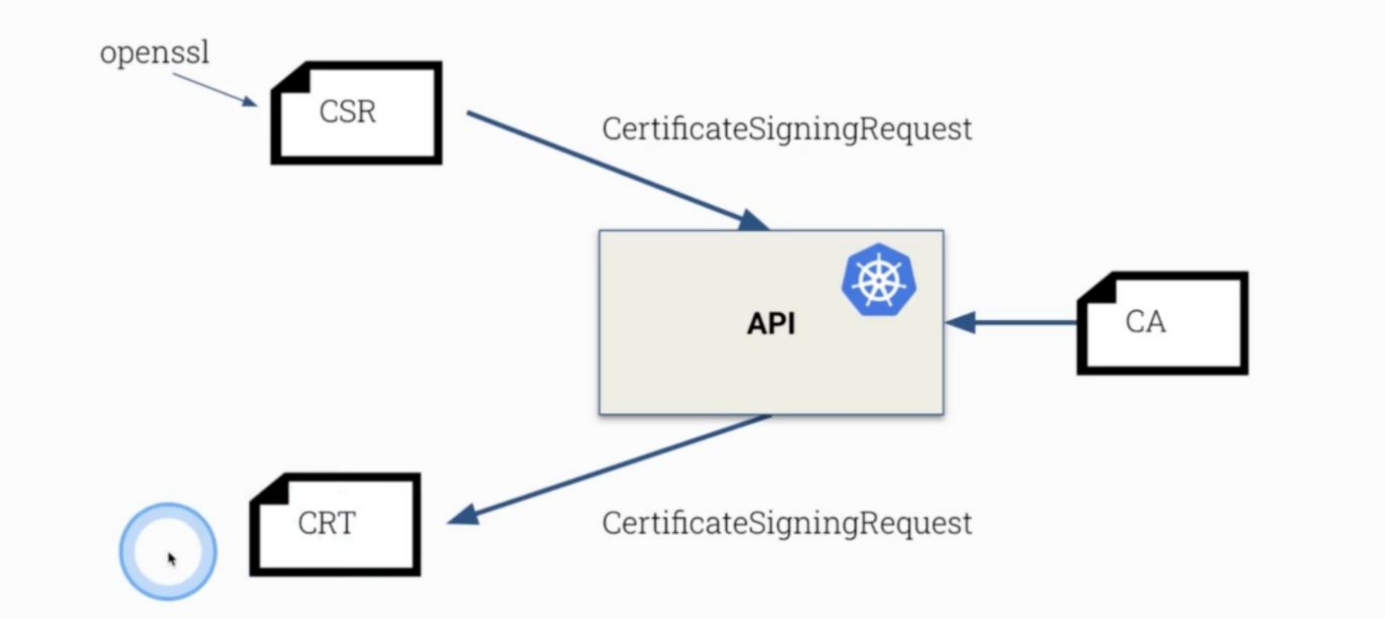
- There is no k8s User resource
- A user is someone with a cert and key

client cert

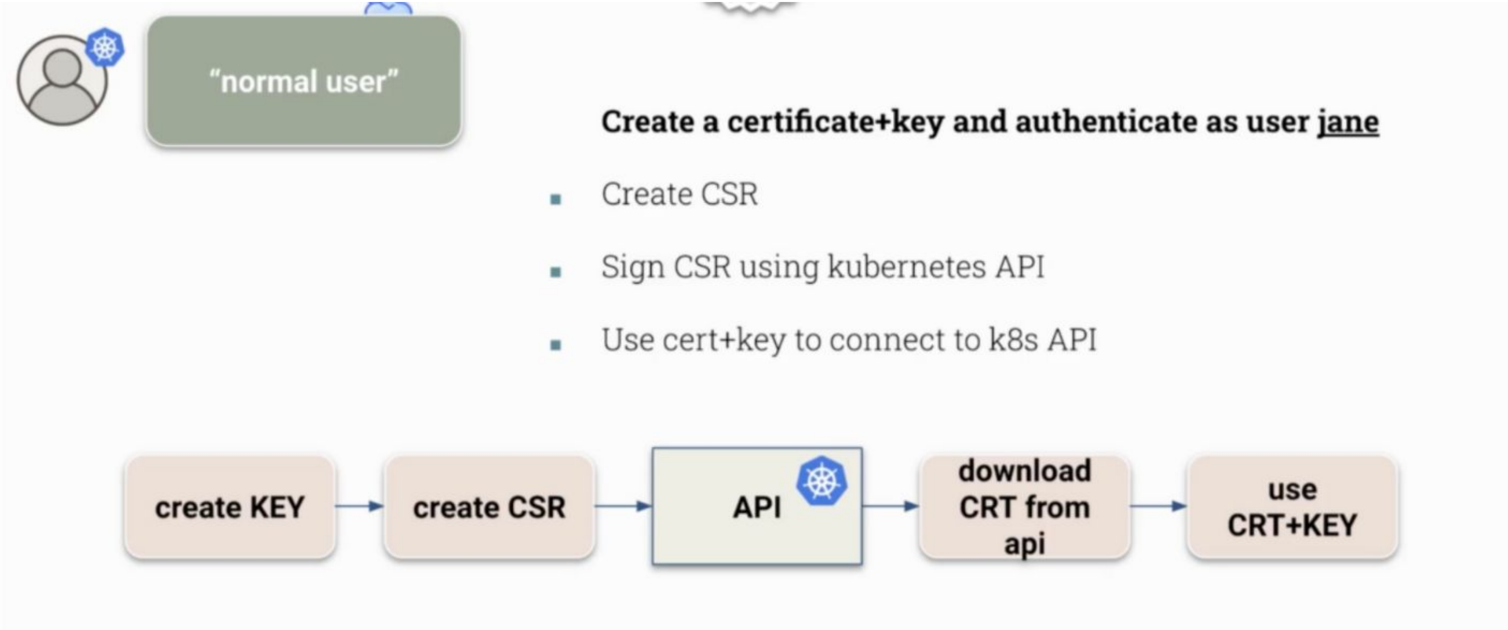
certificate signed by the cluster's
certificate authority (CA)

username under common name
/CN=jane

Certificate Signing



Users and Certificates



“normal user”

Create a certificate+key and authenticate as user jane

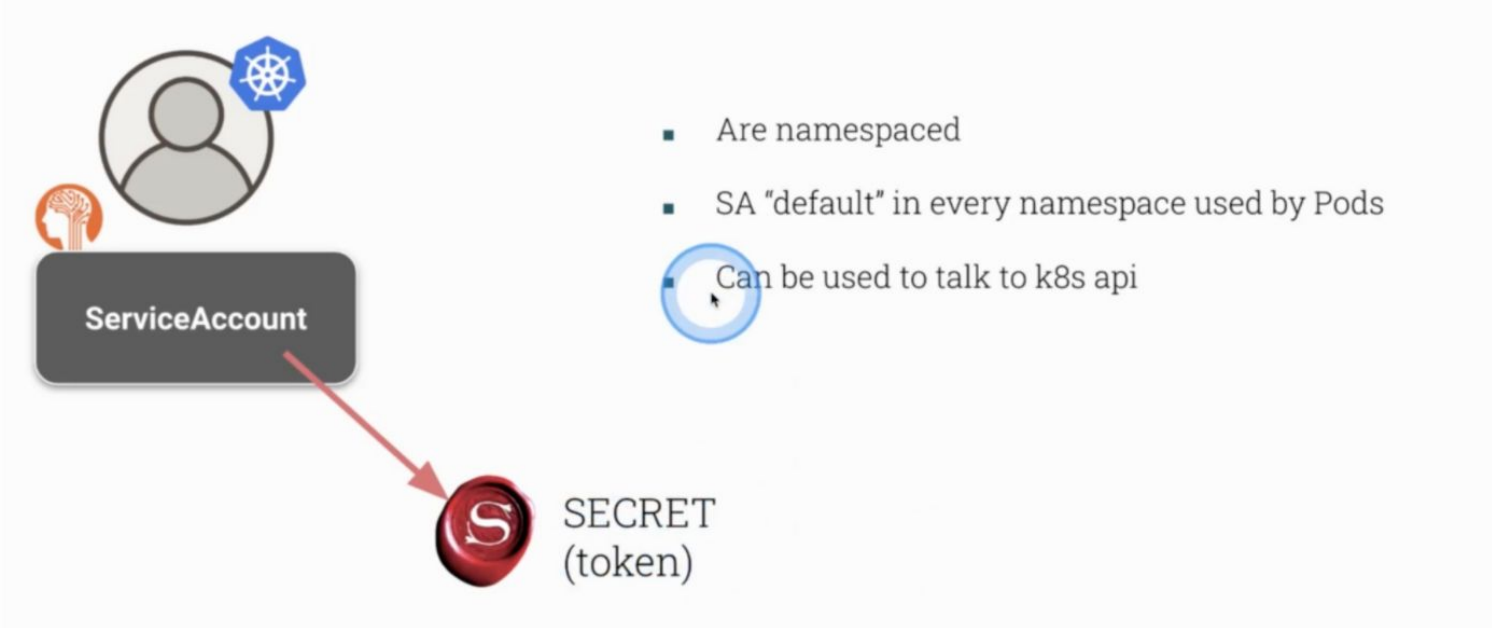
- Create CSR
- Sign CSR using kubernetes API
- Use cert+key to connect to k8s API

create KEY → create CSR → API → download CRT from api → use CRT+KEY

User Certificate Leak - Actions to follow

- There is no way to invalidate a certificate
- If a certificate is leaked,
  - Remove all access associated with the cert
  - Username should not be used again until the cert is expired
  - Create a new CA and re-issue all the certs

Service Accounts



ServiceAccount

SECRET
(token)

- Are namespaced
- SA "default" in every namespace used by Pods
- Can be used to talk to k8s api

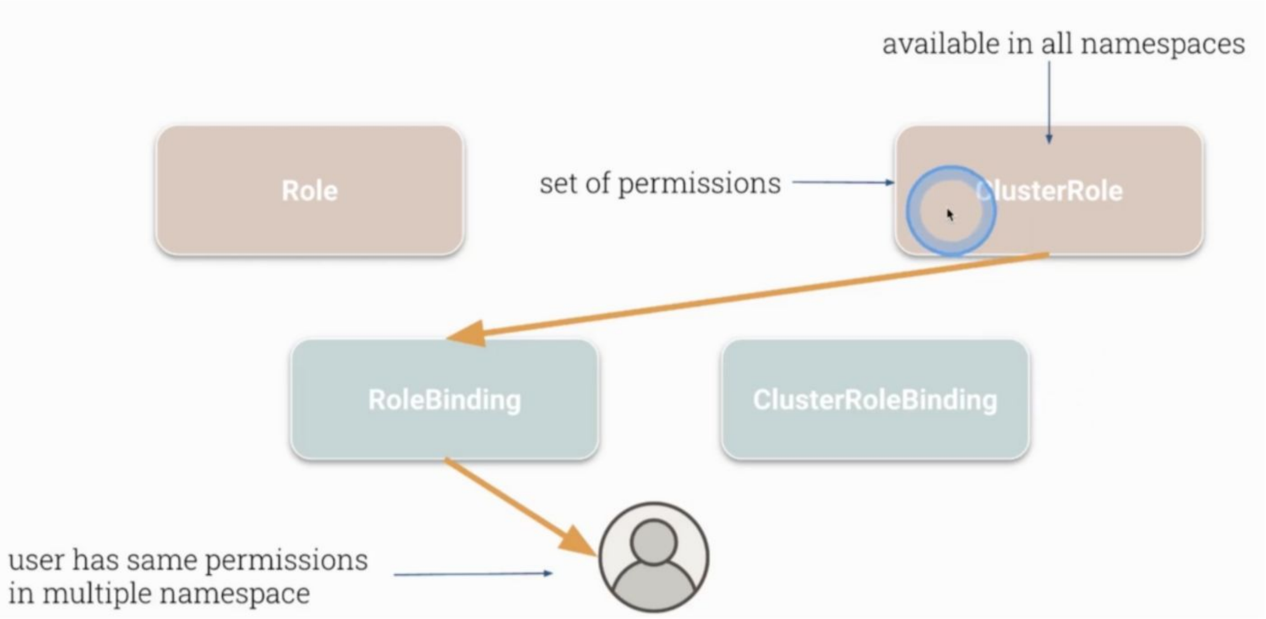
Roles

ClusterRoles - Available Globally to cluster

Role - Available to a namespace only

Role Bindings

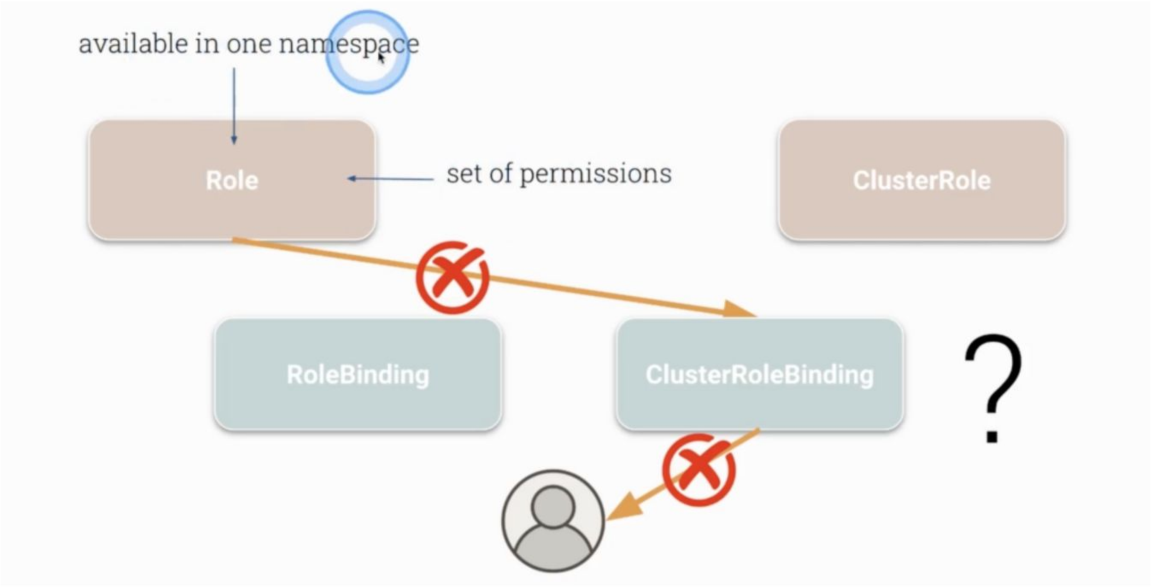
Cluster role and a role can both be binded by a role binding to a namespace



available in all namespaces
Role
set of permissions
ClusterRole
RoleBinding
ClusterRoleBinding
user has same permissions in multiple namespace

Cluster Role binding

Can only bind cluster roles and binding would give cluster wide permissions



Upgrading Kubernetes

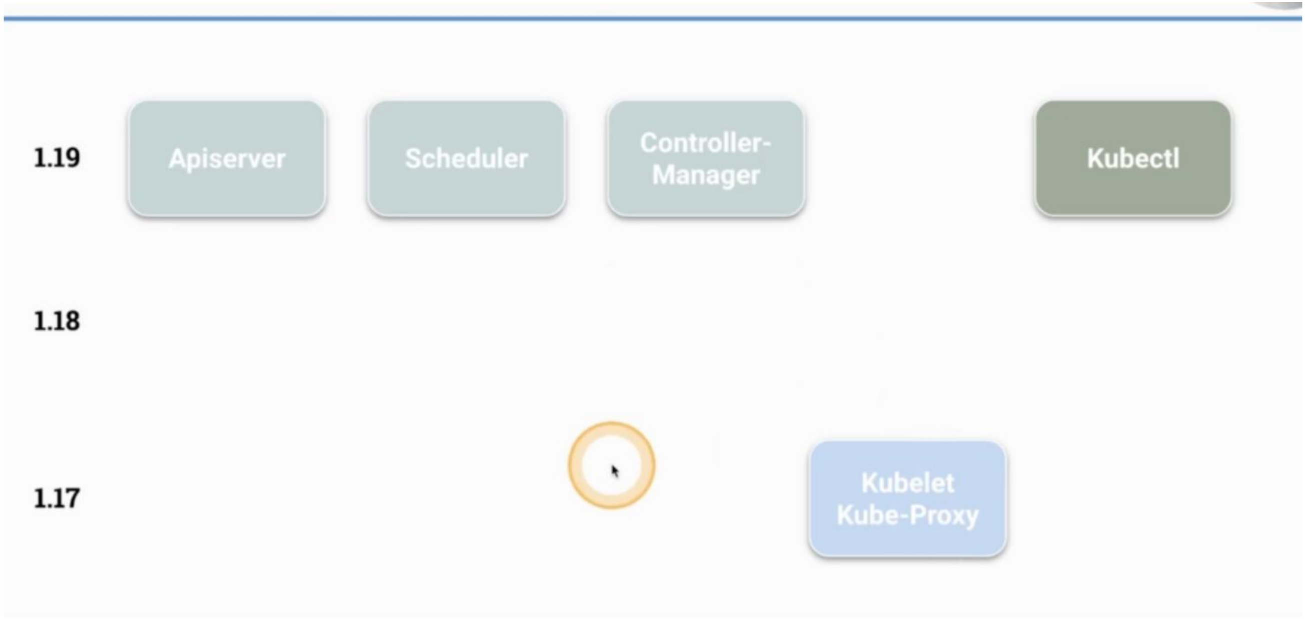
Upgrade Process

- Upgrade master
  - Kubectl drain master --ignore-daemonsets
  - Update kubeadm, kubelet and kubectl
  - Kubeadm upgrade plan
  - Kubeadm upgrade apply
  - Kubectl uncordon master
- Then worker nodes
  - Kubectl drain worker --ignore-daemonsets
  - Update kubeadm
  - Kubeadm upgrade node
  - Update kubelet

Possible different versions

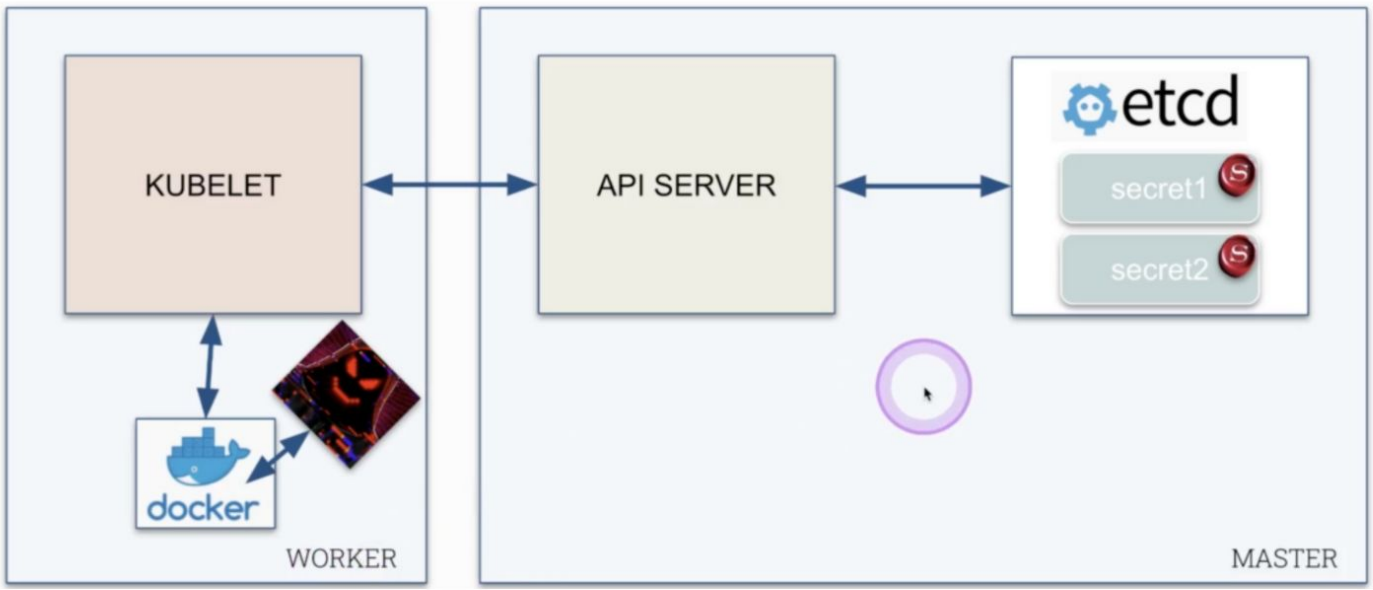
Kubelet can be 2 minor versions under the API server.

But as a rule of thumb, always stick to same versions

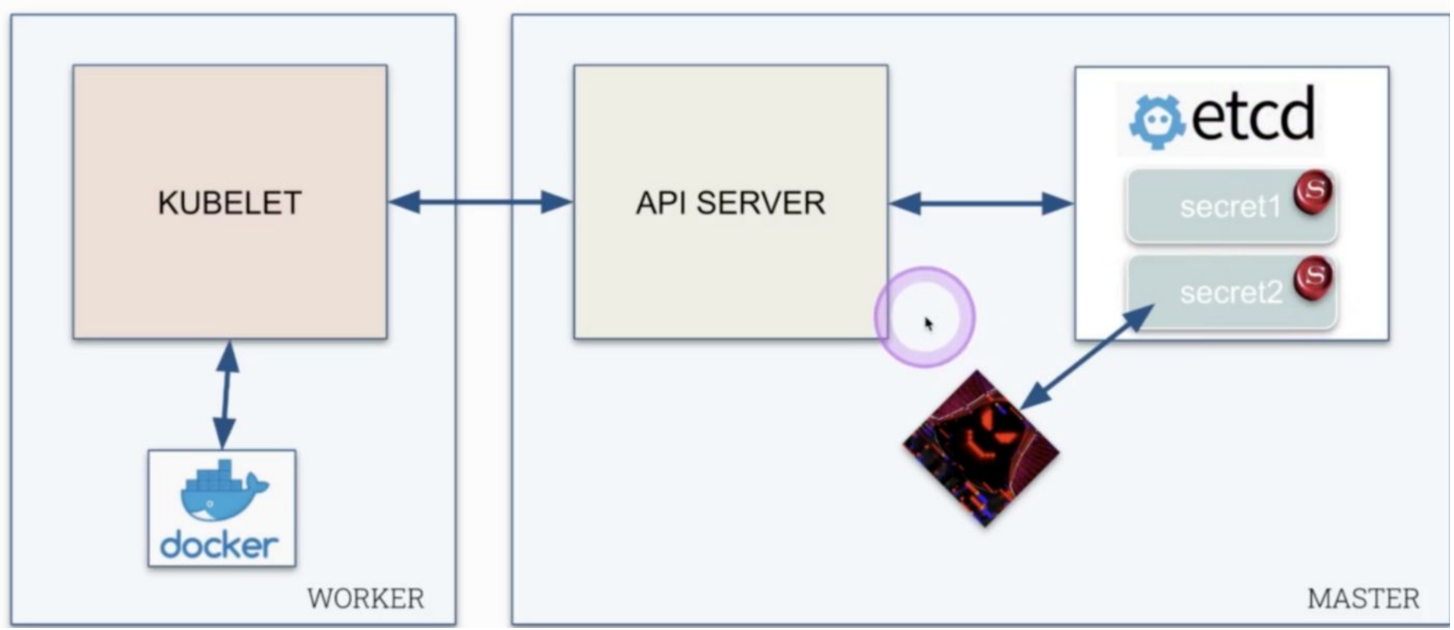


System Hardening

Access Secrets from Docker



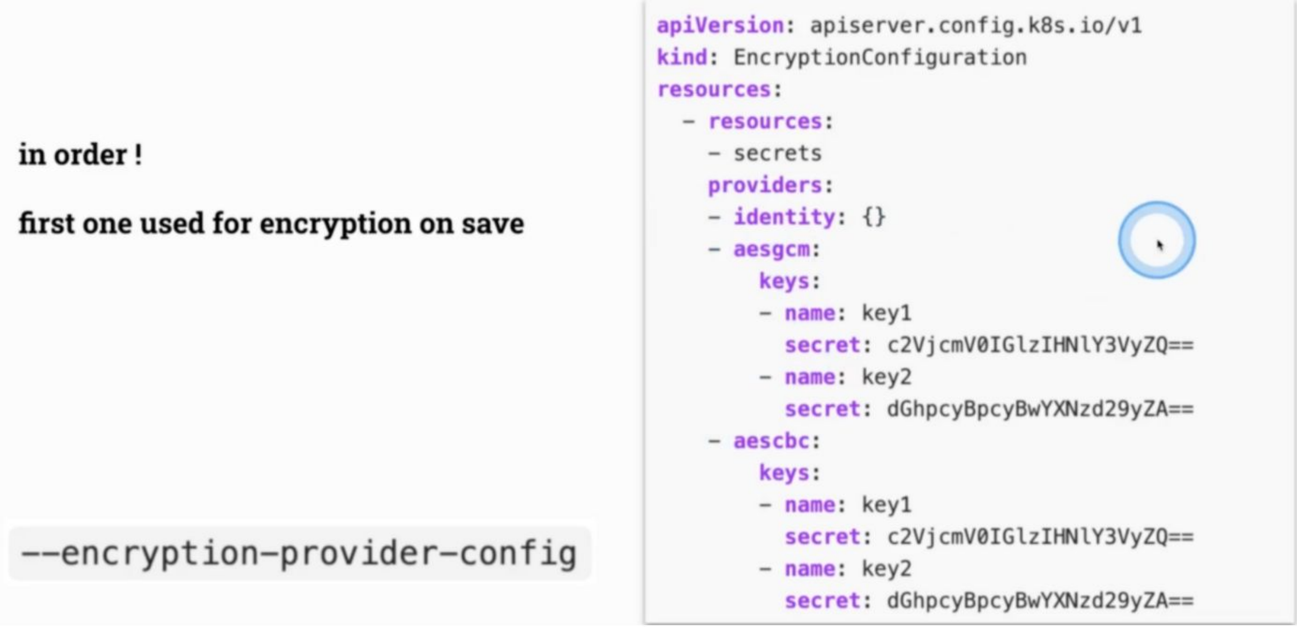
Access Secrets from ETCD



Encrypt ETCD

ALL new secrets are stored un-encrypted because identity is at first

Put identity at the bottom to make sure new secrets are encrypted



in order !

first one used for encryption on save

```
--encryption-provider-config
```

```
apiVersion: apiserver.config.k8s.io/v1
kind: EncryptionConfiguration
resources:
  - resources:
    - secrets
    providers:
    - identity: {}
    - aesgcm:
        keys:
        - name: key1
          secret: c2VjcmV0IGlzIHNlY3VyZQ==
        - name: key2
          secret: dGhpcyBpcyBwYXNzd29yZA==
    - aescbc:
        keys:
        - name: key1
          secret: c2VjcmV0IGlzIHNlY3VyZQ==
        - name: key2
          secret: dGhpcyBpcyBwYXNzd29yZA==
```

Upgrade all Secrets

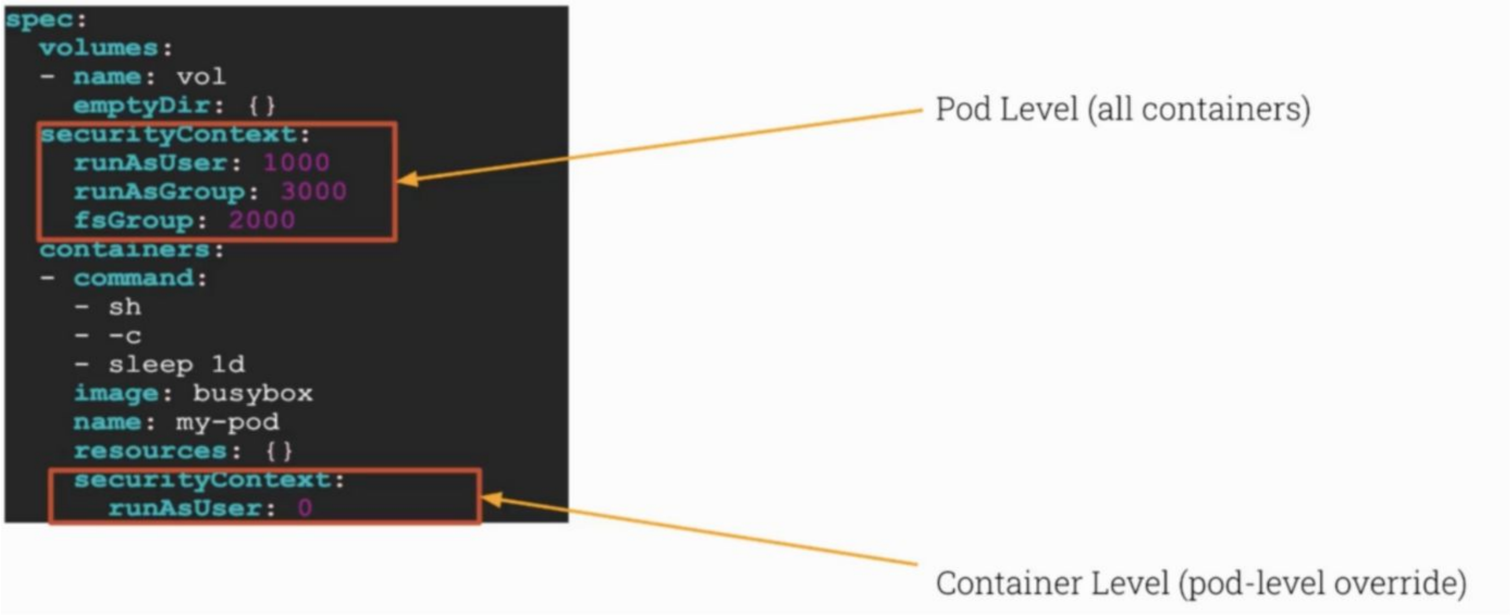
```
kubectl get secrets -A -o json | kubectl replace -f -
```

Generate Encryption Key

```
echo -n password1212212121 | base64

# edit api server and pass

--encryption-provider-config=/etc/kubernetes/etcd/ec.yaml

## mount a hostpath volume and a container mount for the ec. yaml
```

Minimize Micro Service Vulnerabilities

Security Contexts



```
spec:
  volumes:
  - name: vol
    emptyDir: {}
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  containers:
  - command:
    - sh
    - -c
    - sleep 1d
    image: busybox
    name: my-pod
    resources: {}
    securityContext:
      runAsUser: 0
```

Pod Level (all containers)
Container Level (pod-level override)

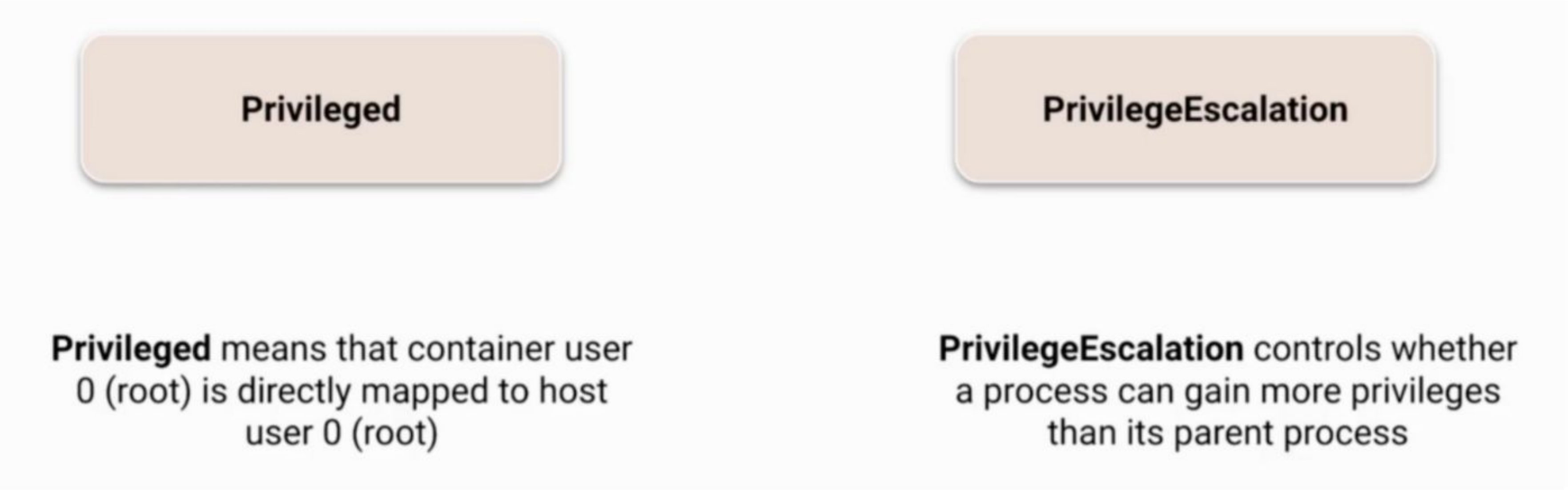
Privileged Containers

- Privileged means container uid 0 is mapped to root user uid 0 of the host machine
- By default, containers run unprivileged

Privilege Escalation

- Privilege Escalation enables a process to gain more privileges than its parent process.
- By default Kubernetes allows privilege escalation

Privilege Escalation



Privileged

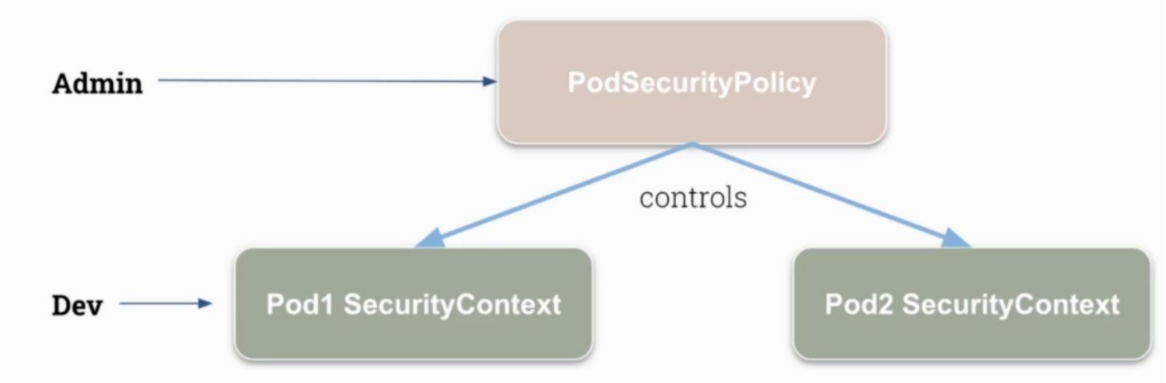
PrivilegeEscalation

Privileged means that container user 0 (root) is directly mapped to host user 0 (root)

PrivilegeEscalation controls whether a process can gain more privileges than its parent process

Pod Security Policies

- Cluster level resource
- Controls under which security contexts the pod should run
- Needs to be enabled by Admission Controller
- Pod should be able to see PodSecurityPolicy with RBAC in order to create it



Enable Pod Security Policies

- Create PSP Resource first
- Add RBAC and necessary service accounts
- THEN go update kubernetes API Server flag --enable-admission-plugins=PodSecurityPolicy

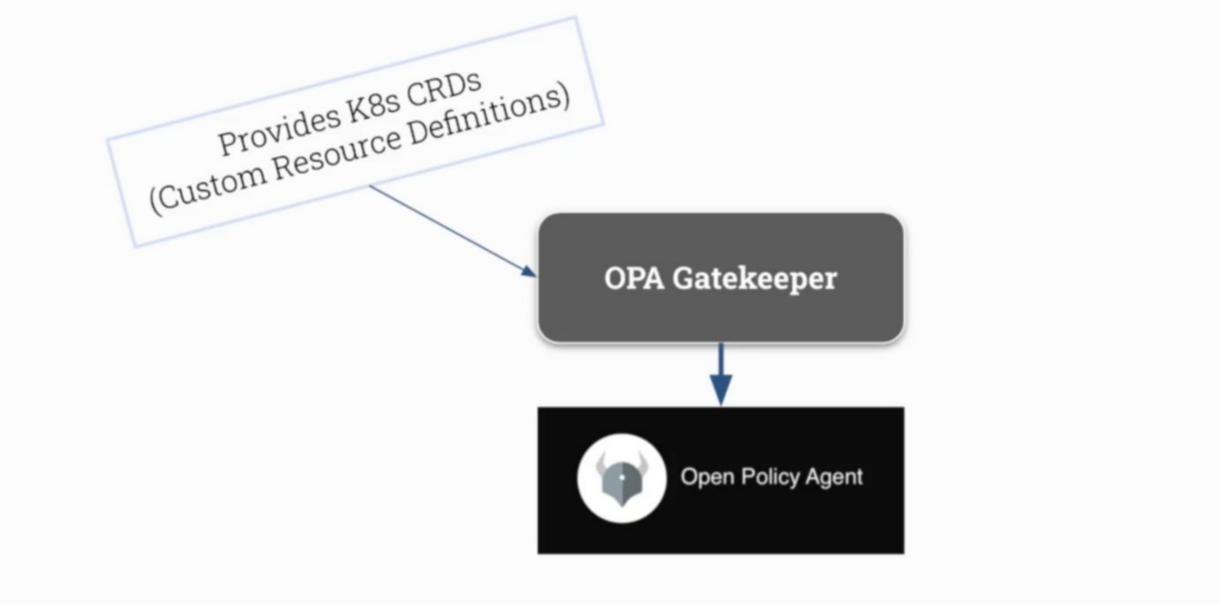
```
k create clusterrole psp-access --verb=use --resource=podsecuritypolicies
k create rolebinding psp-access-binding --clusterrole=psp-access
--serviceaccount=default:default
```

Open Policy Agent

Open policy agent is an open source, general purpose policy engine that enables unified, context aware policy enforcement across entire stack

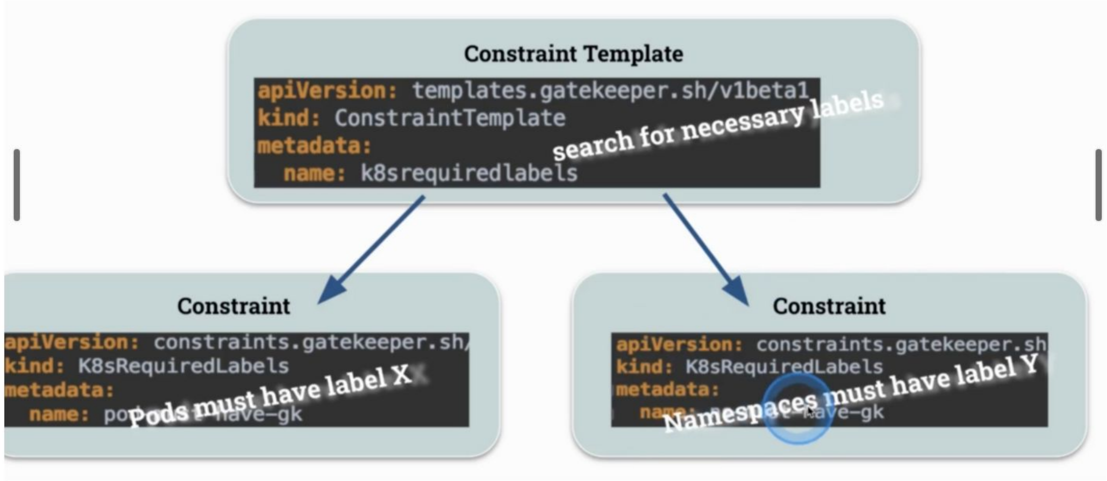
- Not Kubernetes specific
- Easy to implement
- Works with JSON and YAML
- Use Admission Controllers in Kubernetes
- Does not know concepts like Pods and Deployments

OPA Gatekeeper



OPA CRDS

As you can see from below implementation, The NAME given to constraintTemplate is the CRD kind for the constraint. OPA Gatekeeper creates CRD resources dynamically and implements templates we define.



Constraint Template

```
apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: k8srequiredlabels
```

Constraint

```
apiVersion: constraints.gatekeeper.sh/
kind: K8sRequiredLabels
metadata:
  name: po     have-gk
```

Constraint

```
apiVersion: constraints.gatekeeper.sh
kind: K8sRequiredLabels
metadata:
  name:      have-gk
```

OPA Installation

These support Dynamic Admission Controllers, which means, you don't have to edit the API server admission webhook list every time.

validatingadmissionwebhook - Validates an object

mutatingadmissionwebhook - injects content into an object at creation

Practicals
https://github.com/nilesh93/cks-course-environment/tree/master/course-content/opa

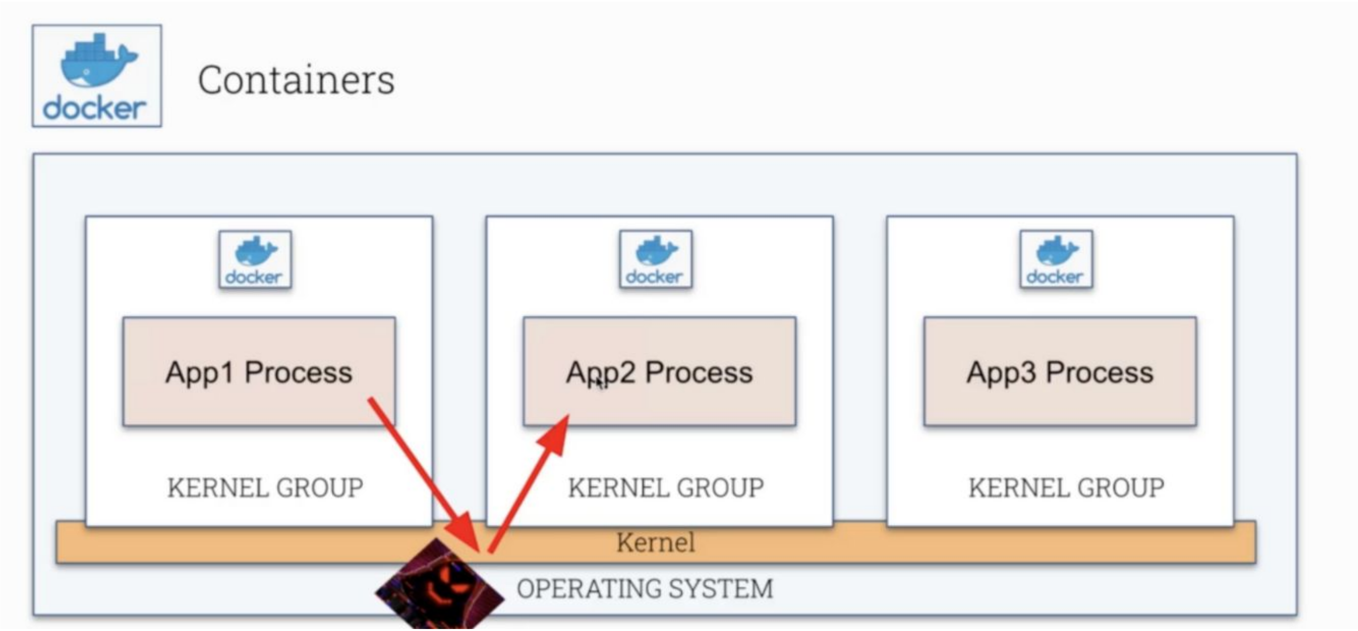
Additional Resources
https://github.com/BouweCeunen/gatekeeper-policies
https://www.youtube.com/watch?v=RDWndems-sk&feature=emb_title

Kubernetes Secrets

- Secrets are similar to Configmaps
- Secrets are stored in Kubernetes as a base64 encoded string
- Secrets cannot be shared across namespaces
- Best practice is to mount secrets as files and environment variable injection is not recommended

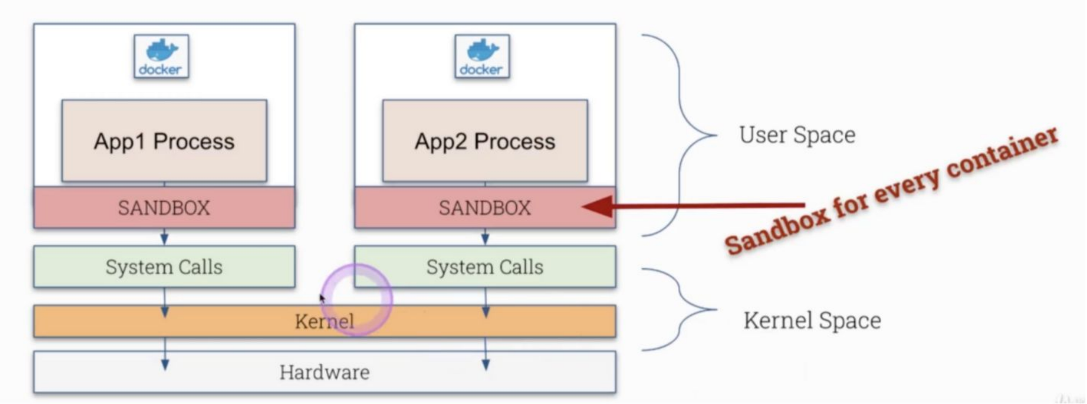
Additional Reading
https://kubernetes.io/docs/concepts/configuration/secret/

Container Runtime Attack Surface



What is a Sandbox

- Playground when implementing an API
- Simulated Test environment
- Development Server
- Security Layer to reduce Attack Surface

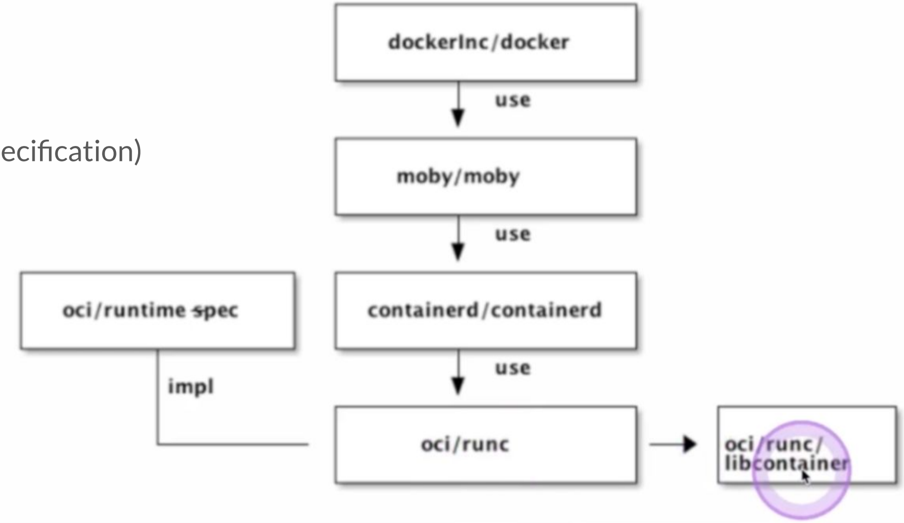


Sandboxes disadvantages

- More resources needed
- Better for smaller containers
- Not good for syscall heavy workloads
- No direct access to Hardware

Open Container Initiative

- Linux Foundation designs and Spec for Open standards for virtualization
- Specification
  - Runtime, image, distribution
- Runtime
  - Runc (container runtime that invokes the specification)



dockerInc/docker

use

moby/moby

use

oci/runtime spec

containerd/containerd

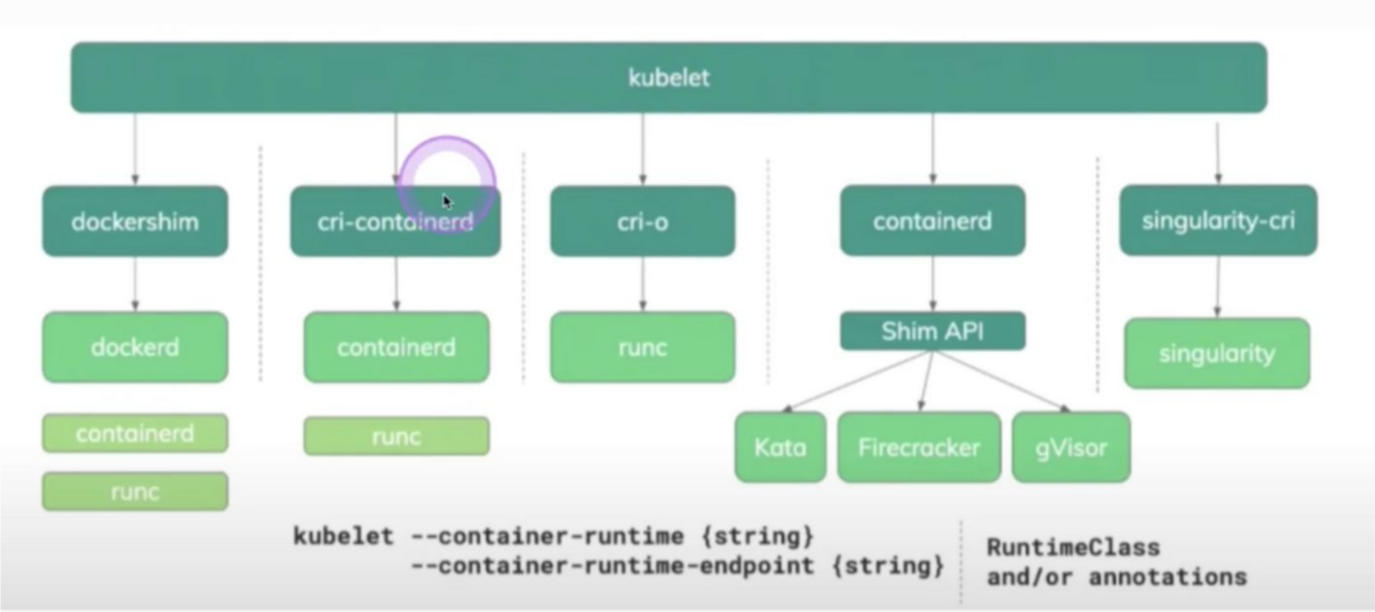
impl

use

oci/runc

oci/runc/
libcontainer

Kubernetes Runtimes



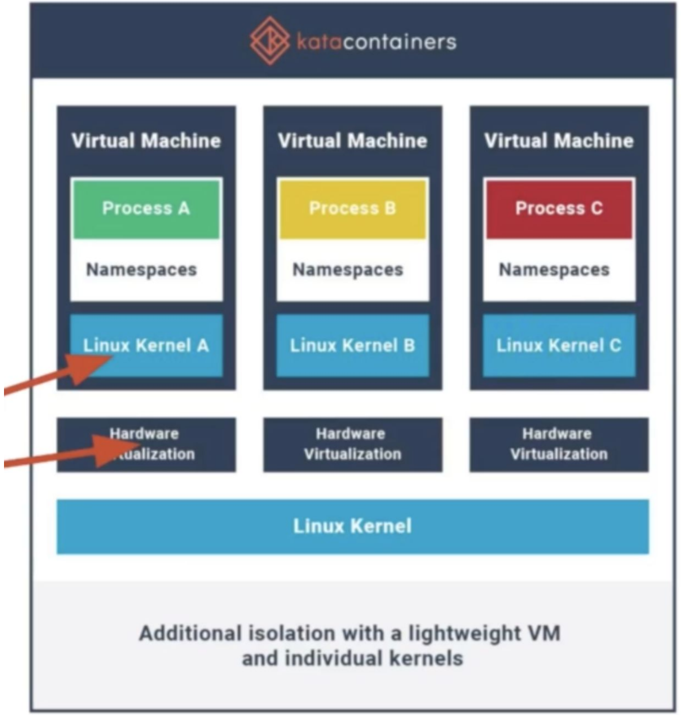
```
kubelet --container-runtime {string}
        --container-runtime-endpoint {string}
```

```
RuntimeClass
and/or annotations
```

Kata Containers

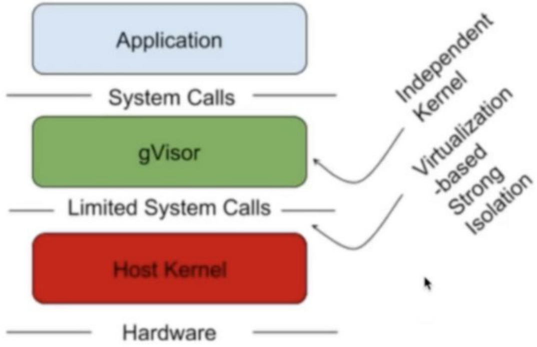
- Strong separation layer
- Every container is running in a private VM
- QEMU by default.

QEMU is not supported in cloud providers, might have to use other virtualisation techniques in cloud provided VMs



gVisor

- Additional layer of separation
- Not Hypervisor based
- Runtime called runsc
- Runs is user space separated from kernel
- Simulates kernel syscalls with limited functionality

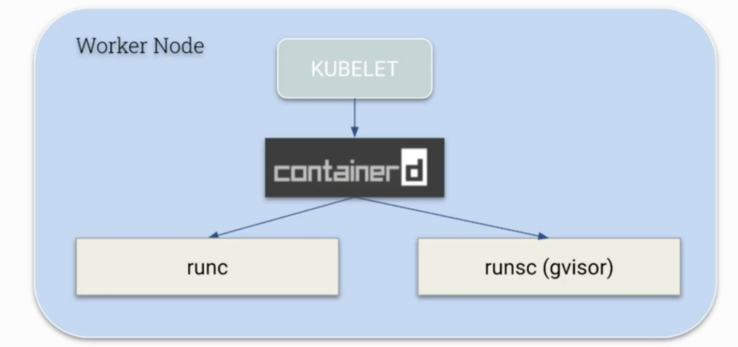


Using gVisor in Kubernetes

```
bash <(curl -s
https://raw.githubusercontent.com/nilesh93/cks-course-environment/master/course-content/microservice-vulnerabilities/container-runtimes/gvisor/install_gvisor.sh)
```

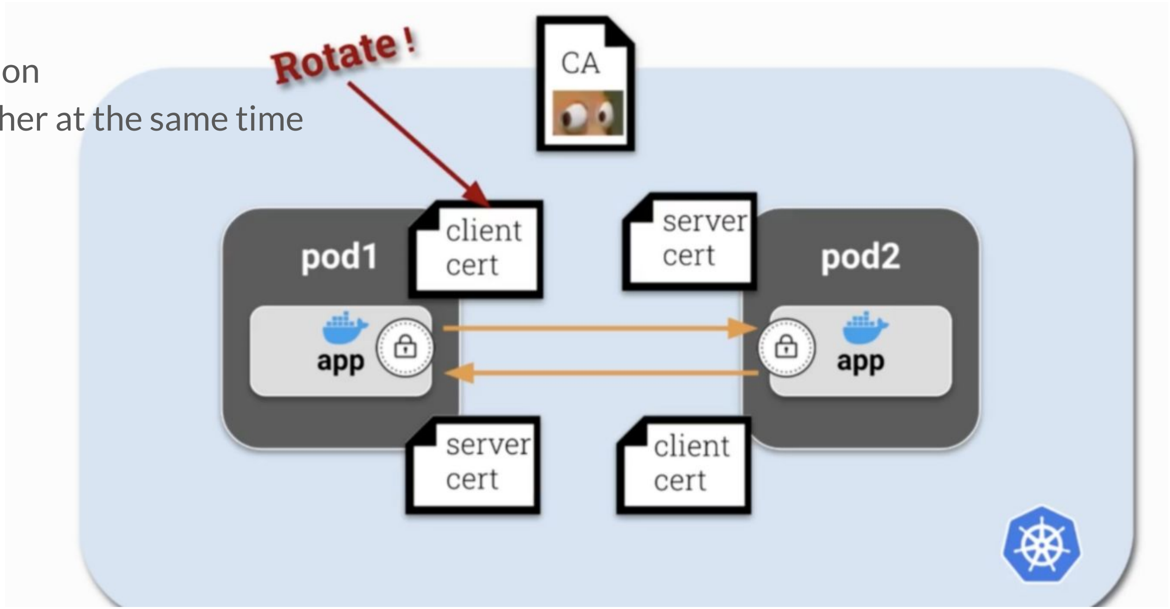
This script actually install containerd and configures kubelet to use containerd instead of docker

After that create a runtimeclass in Kubernetes https://kubernetes.io/docs/concepts/containers/runtime-class/

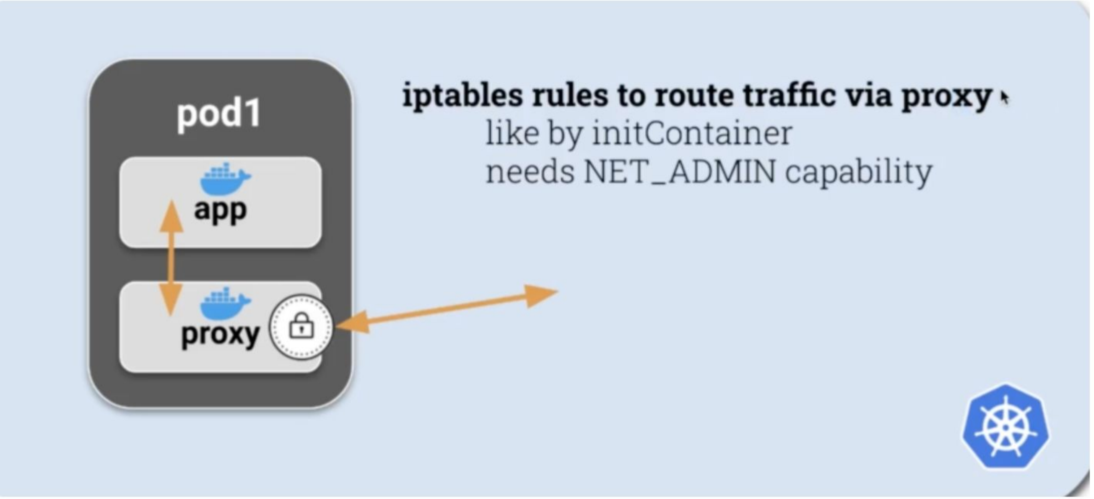


mTLS

- Mutual Authentication
- Two way bilateral authentication
- 2 Parties authenticate each other at the same time



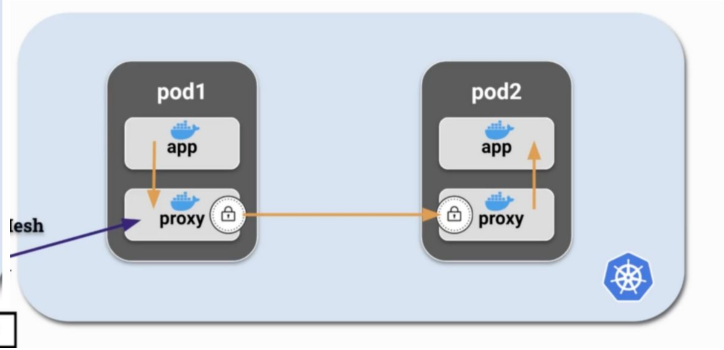
Service Mesh



iptables rules to route traffic via proxy

like by initContainer

needs NET_ADMIN capability

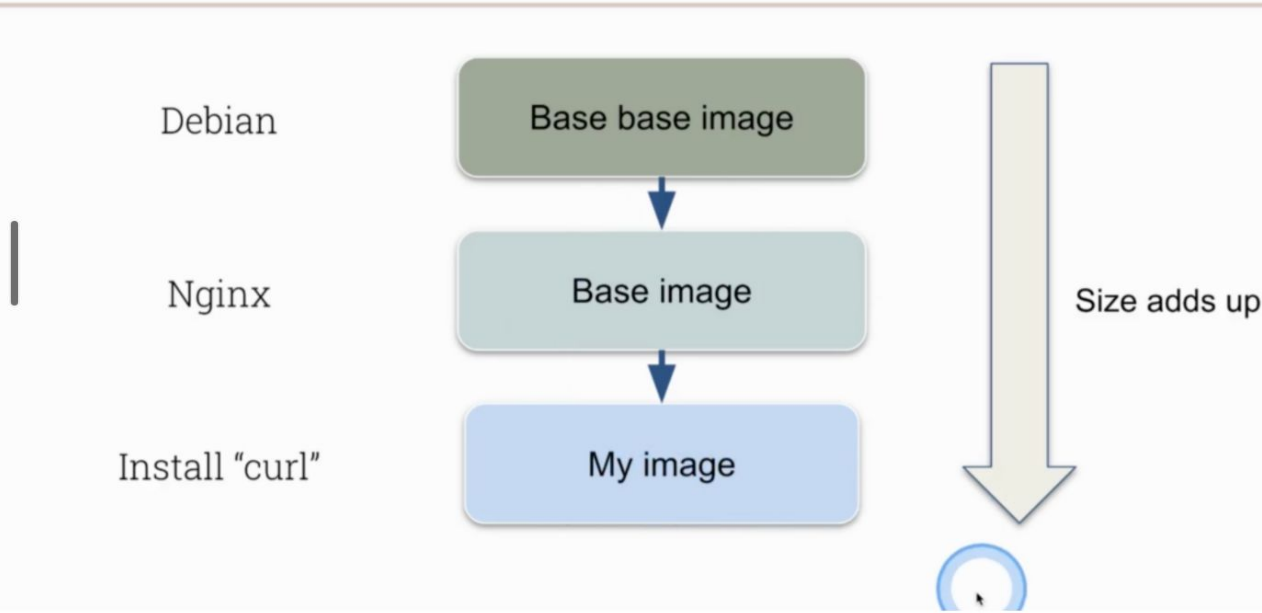


Supply Chain Security

Reduce Image Footprint

Multi Stage builds reduce image footprint

Use alpine images as much as possible



Harden Images

- Don’t use latest tag and use specific images
- Always go for official images
- Don’t run as root

```
RUN addgroup -S appgroup && adduser -S appuser -G appgroup -h /home/appuser
# copy the executable to /home/appuser instead of /app

USER appuser
```

- Make file system read only

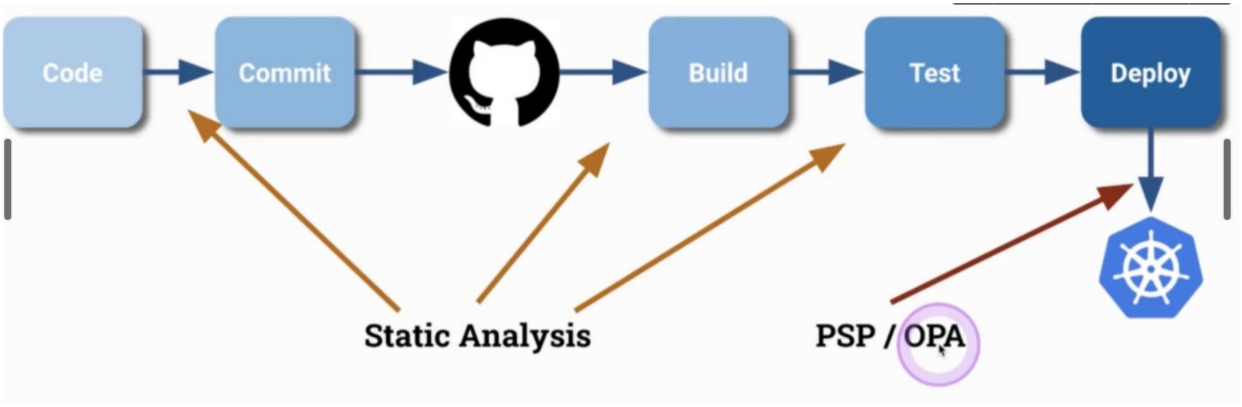
```
RUN chmod a-w /etc
```

- Remove shell access

```
# add this in the last step
RUN rm -rf /bin/*
```

Static Analysis

- Enforce rules
- Check against rules
- Look at source code files



Kubesec

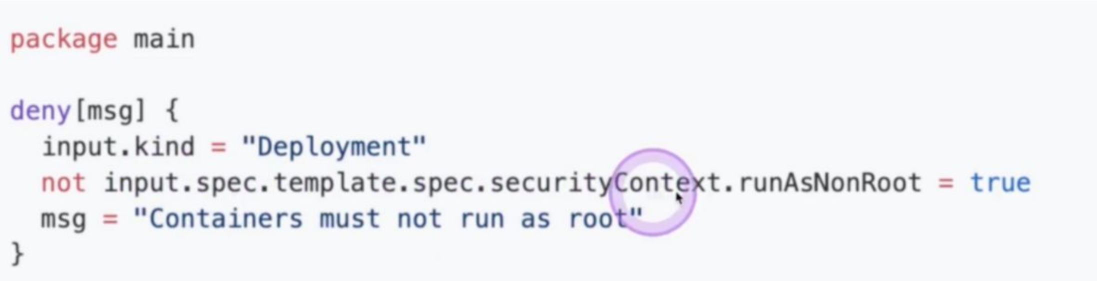
- Security risk analysis for Kubernetes configs
- Opensource
- Fixed set of rules with security best practices

```
docker run -i kubesec/kubesec:512c5e0 scan /dev/stdin < pod.yaml
```

Conftest - OPA

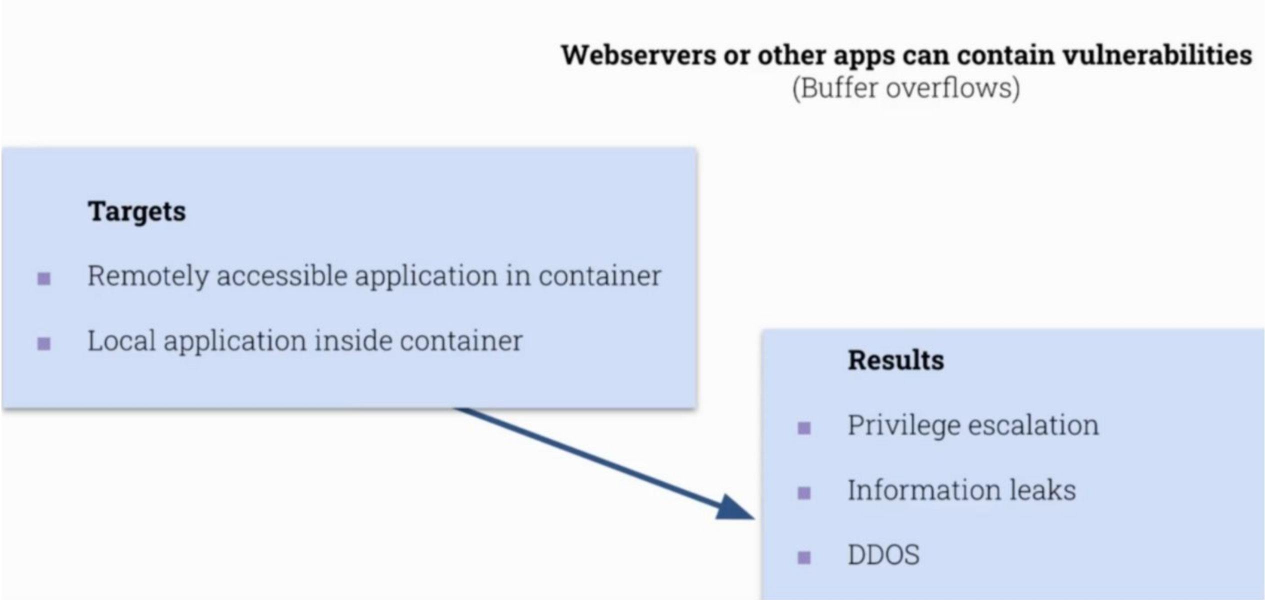
- Part of Open Policy Agent
- Unit test framework for kubernetes resources

```
git clone https://github.com/nilesh93/cks-course-environment.git cd
cks-course-environment/course-content/supply-chain-security/static-analysis/conftest/kubernetes docker
run --rm -v $(pwd):/project openpolicyagent/conftest test deploy.yaml
```



```
package main

deny[msg] {
  input.kind = "Deployment"
  not input.spec.template.spec.securityContext.runAsNonRoot = true
  msg = "Containers must not run as root"
}
```

Image Vulnerability Scanning



Webservers or other apps can contain vulnerabilities
(Buffer overflows)

Targets

- Remotely accessible application in container
- Local application inside container

Results

- Privilege escalation
- Information leaks
- DDOS

Trivy

- Open Source Project
- One of the best lightweight tools developed by aquasec to scan images

```
docker run ghcr.io/aquasecurity/trivy:latest image nginx:latest
```

```
trivy nginx:latest
```

Image Policy Admission Controller

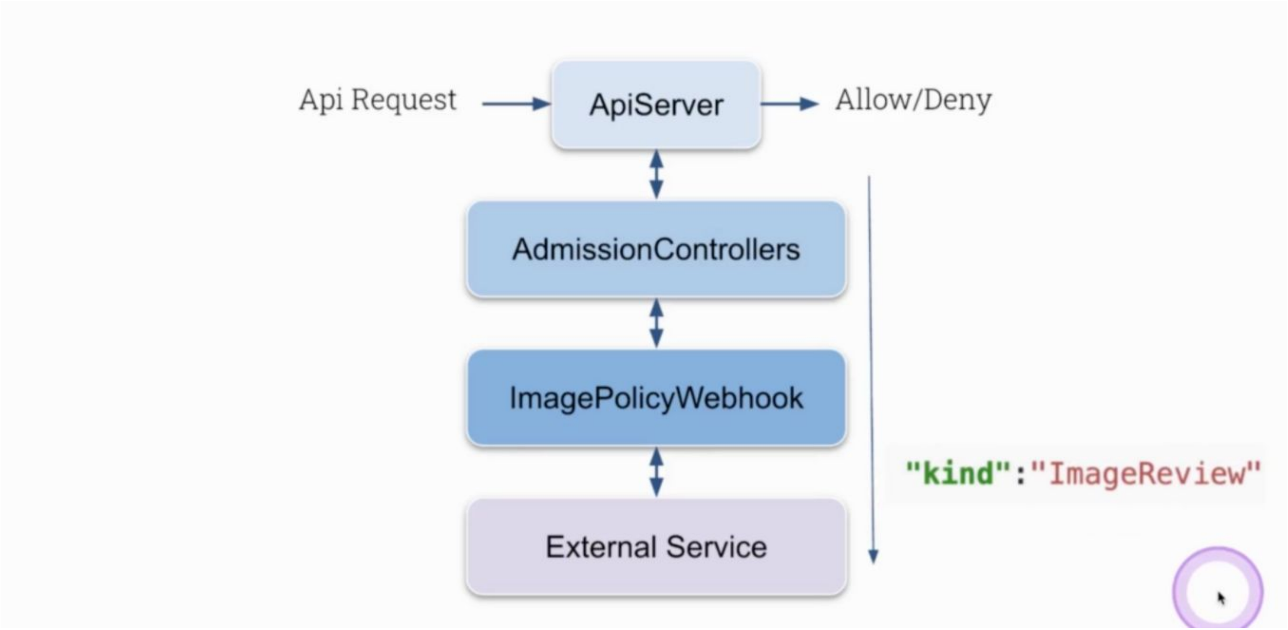
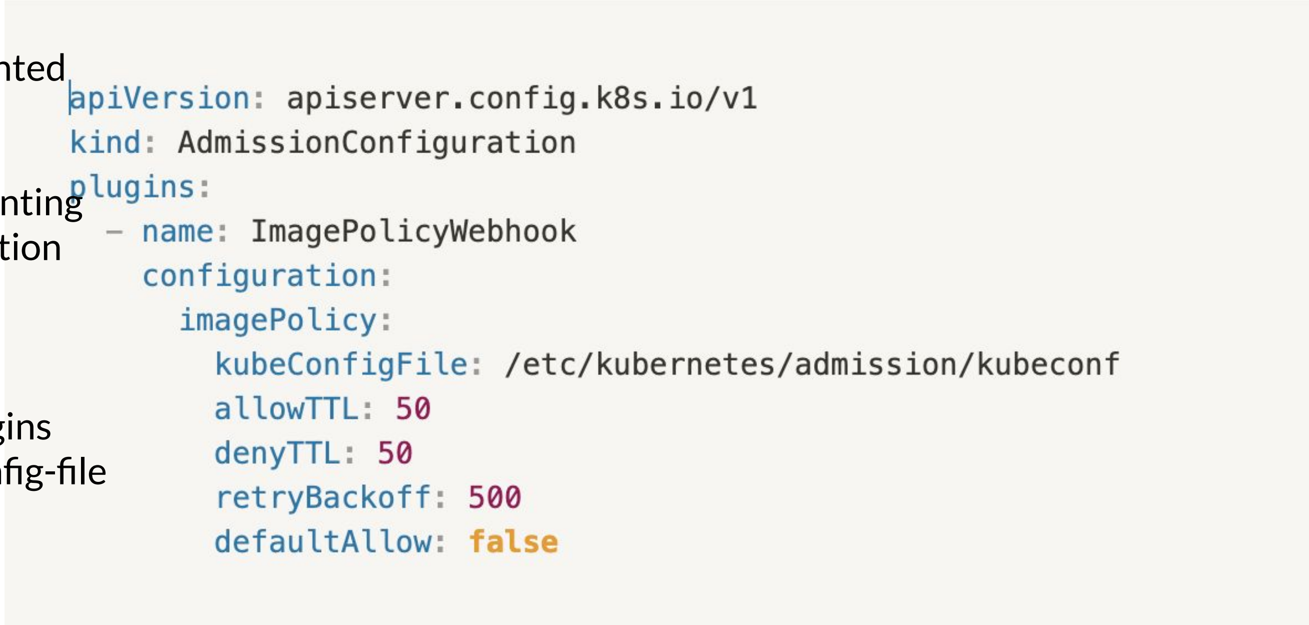


Image Policy Installation

Config needs to be mounted into API Server

KubeConf should be pointing to external image validation service

Enable via
--enable-admission-plugins
--admission-control-config-file



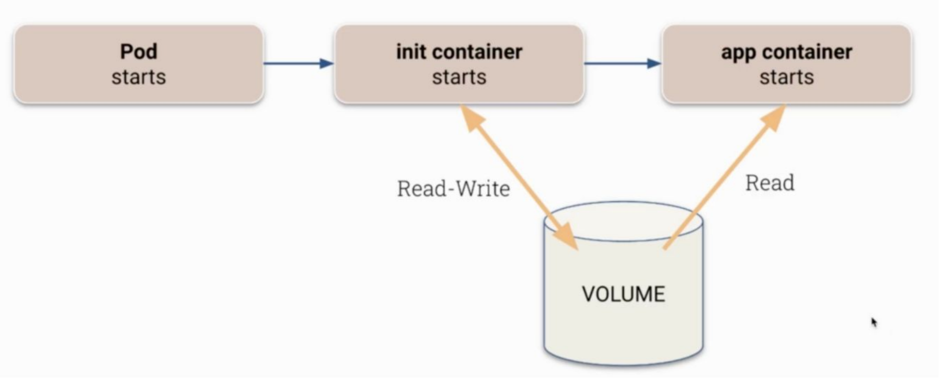
```yaml
apiVersion: apiserver.config.k8s.io/v1
kind: AdmissionConfiguration
plugins:
  - name: ImagePolicyWebhook
    configuration:
      imagePolicy:
        kubeConfigFile: /etc/kubernetes/admission/kubeconf
        allowTTL: 50
        denyTTL: 50
        retryBackoff: 500
        defaultAllow: false
```

Container Immutability

- Remove Bash
- File system read only
- Run as a non root user

All of these can be done on Kubernetes level

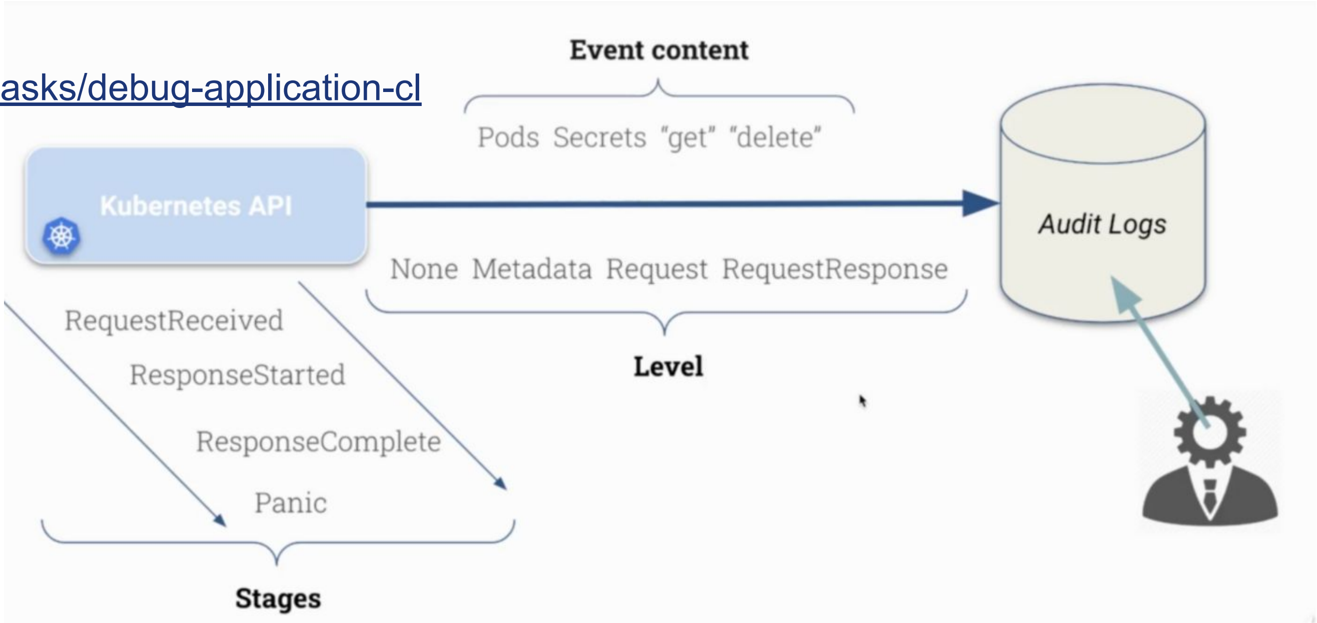
- Writing files - Empty Dir
- Initializing files - init container



Runtime Security

Audit Logs

Additional Reading
https://kubernetes.io/docs/tasks/debug-application-cluster/audit/



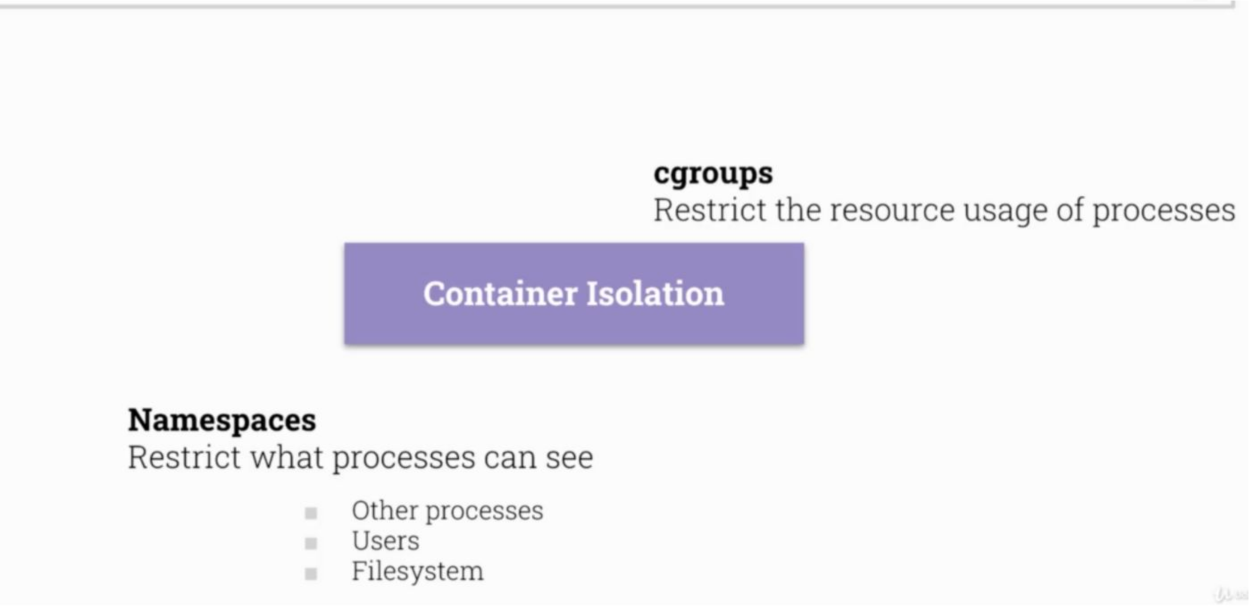
Enable Audit Logs

- Create Audit Policy in master. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/
- Mount Policy to API server
- Add Policy Flags

```
- --audit-policy-file=/etc/kubernetes/audit/policy.yaml
- --audit-log-path=/etc/kubernetes/audit/logs/audit.log
- --audit-log-maxsize=500
- --audit-log-maxbackup=5
```

Additional reading
https://www.youtube.com/watch?v=HXtLTxo30SY&feature=emb_title

Linux Kernel Isolation



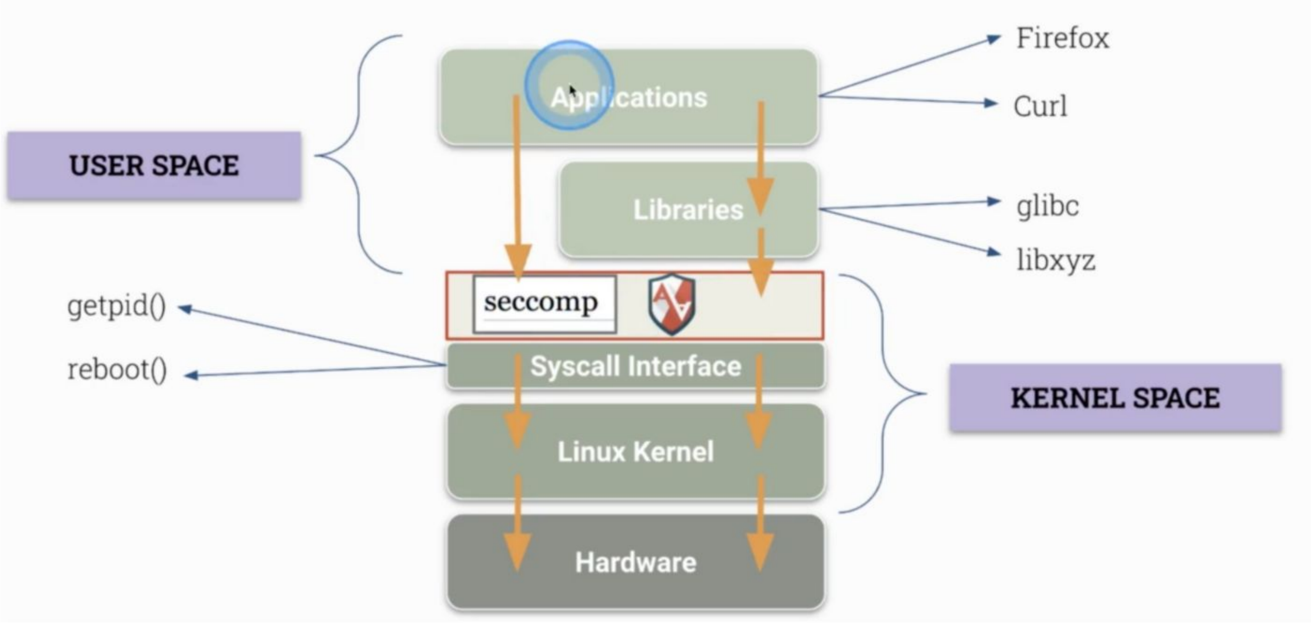
cgroups
Restrict the resource usage of processes

Container Isolation

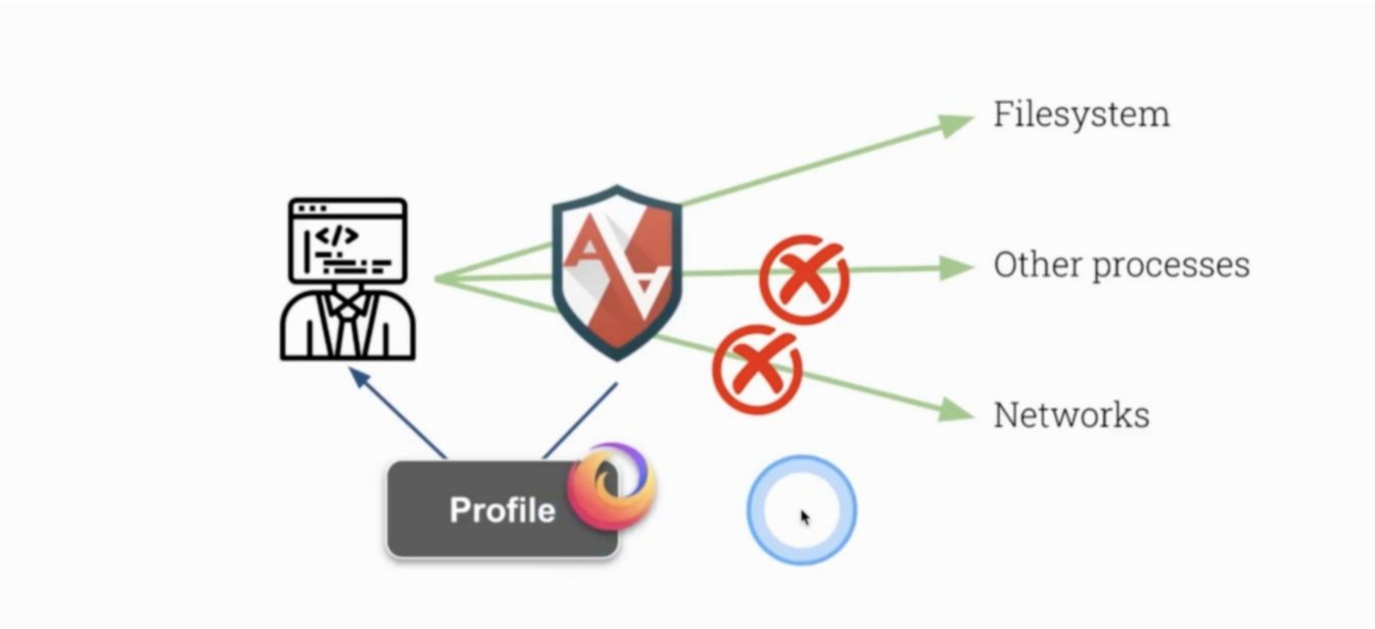
Namespaces
Restrict what processes can see

- Other processes
- Users
- Filesystem

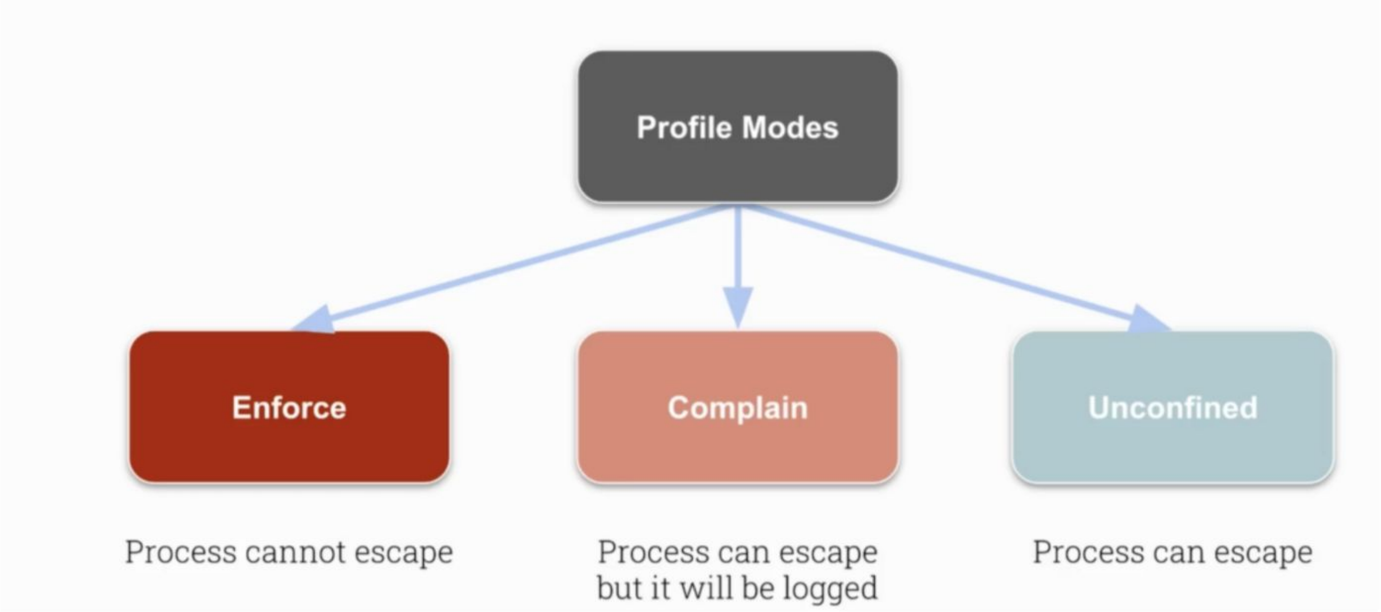
Kernel vs User Space



App Armor



App Armor Profiles



Profile Modes

Enforce

Complain

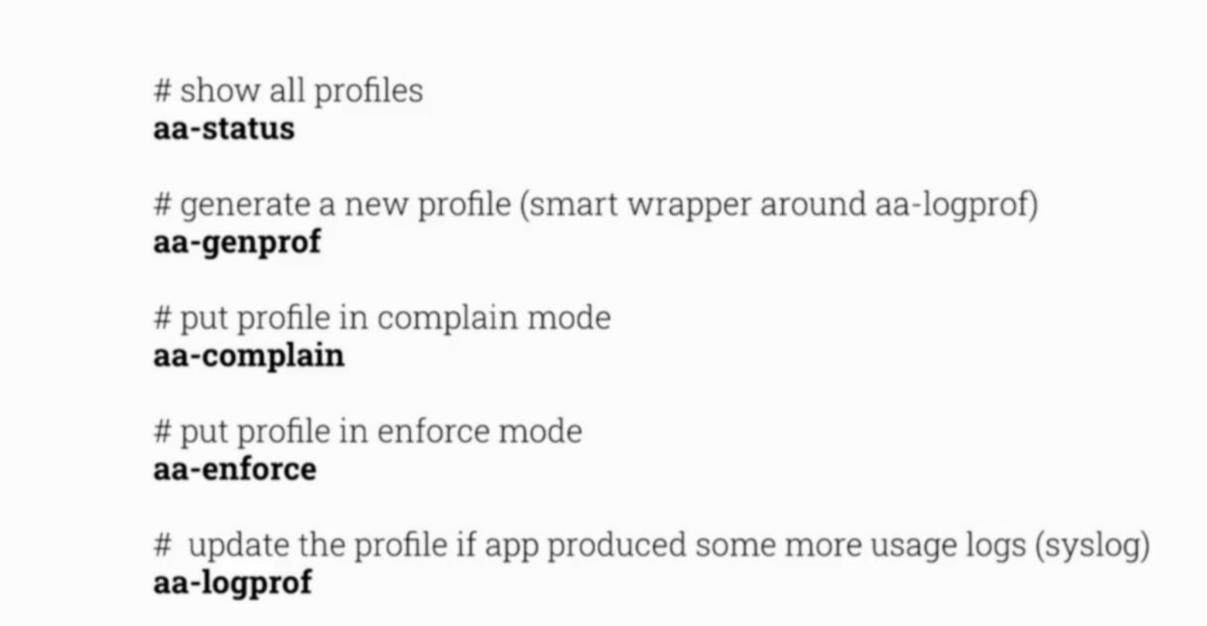
Unconfined

Process cannot escape

Process can escape but it will be logged

Process can escape

App Armor Commands



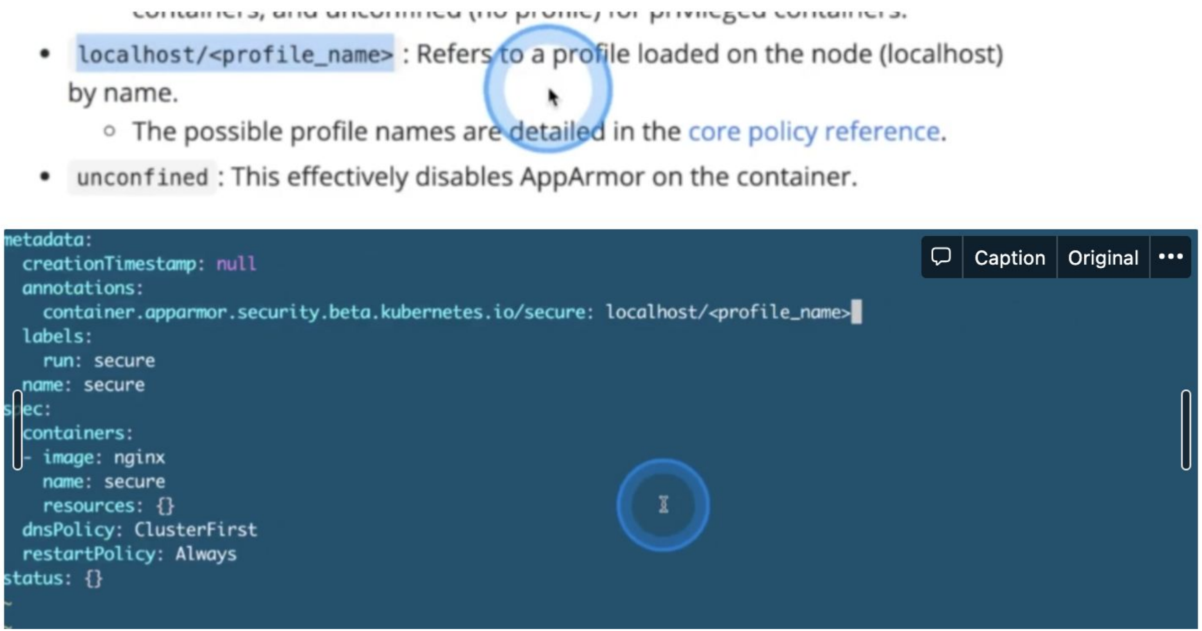
```
# show all profiles
aa-status

# generate a new profile (smart wrapper around aa-logprof)
aa-genprof

# put profile in complain mode
aa-complain

# put profile in enforce mode
aa-enforce

#  update the profile if app produced some more usage logs (syslog)
aa-logprof
```

Generate App Armor Profile

```
apt-get install apparmor-utils
aa-genprof curl


# Run curl and see it doesn't work
# using logprof update the apparmor profile
cd /etc/apparmor.d/ aa-logprof

# Install custom profile
apparmor_parser /etc/apparmor.d/<file-name>
```

Use App Armor with Kubernetes

Additional Reading
https://kubernetes.io/docs/tutorials/clusters/apparmor/



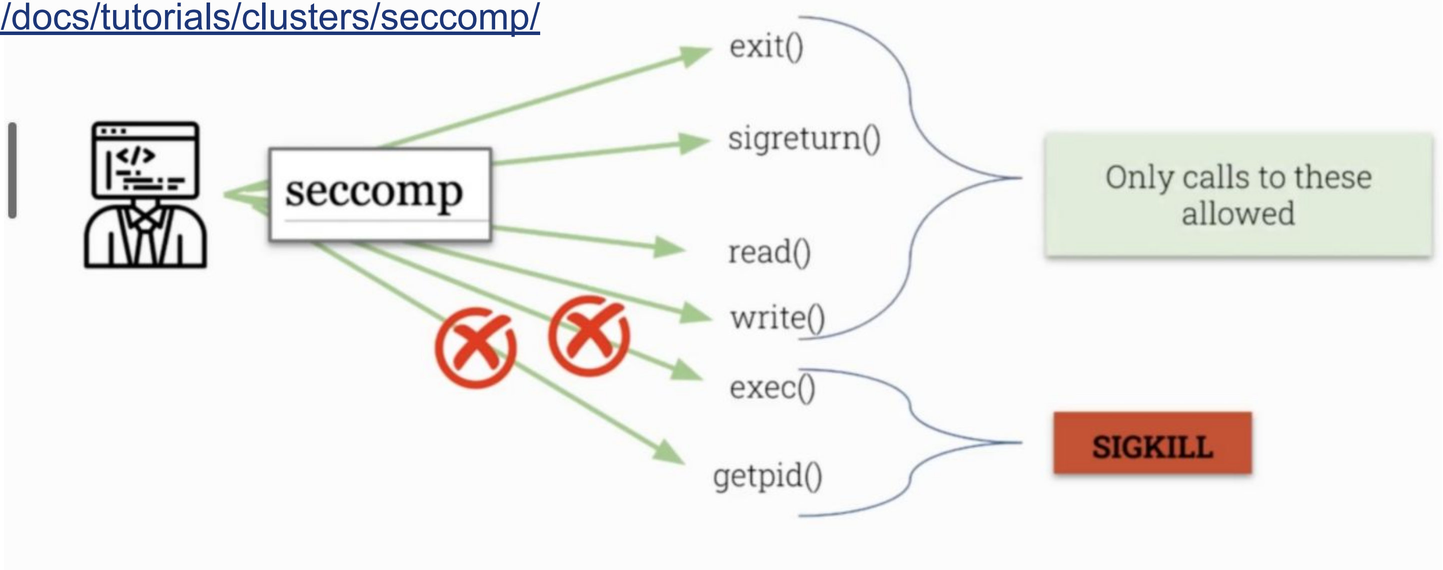
containers, and unconfined (no profile) for privileged containers.

- `localhost/<profile_name>` : Refers to a profile loaded on the node (localhost) by name.
  - The possible profile names are detailed in the core policy reference.
- `unconfined` : This effectively disables AppArmor on the container.

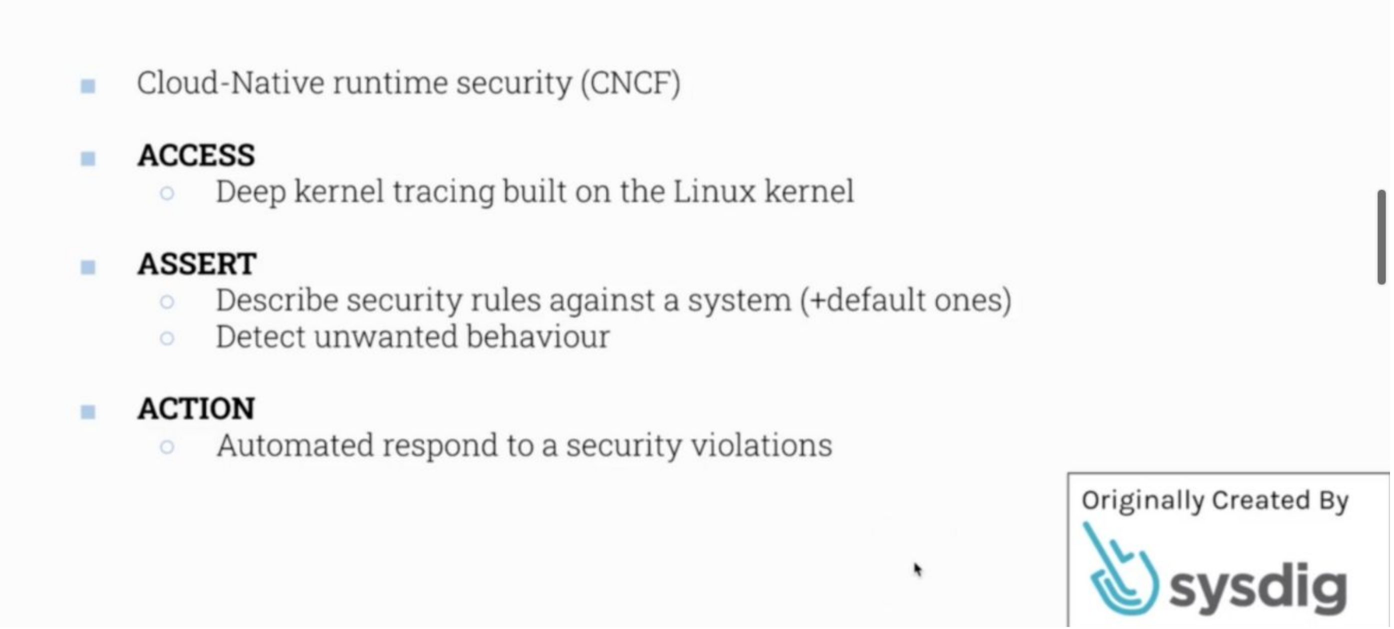
```
metadata:
  creationTimestamp: null
  annotations:
    container.apparmor.security.beta.kubernetes.io/secure: localhost/<profile_name>
  labels:
    run: secure
  name: secure
spec:
  containers:
  - image: nginx
    name: secure
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
~
```

Seccomp

Additional Reading
https://kubernetes.io/docs/tutorials/clusters/seccomp/



Falco - Runtime Scanning



- Cloud-Native runtime security (CNCF)
- **ACCESS**
  - Deep kernel tracing built on the Linux kernel
- **ASSERT**
  - Describe security rules against a system (+default ones)
  - Detect unwanted behaviour
- **ACTION**
  - Automated respond to a security violations

Originally Created By

sysdig

Install Falco

- Needs to be installed on all nodes. (standalone, daemonset)
- All configs are at /etc/falco

```
# install falco

curl -s https://falco.org/repo/falcosecurity-3672BA8F.asc | apt-key add - echo "deb
https://dl.bintray.com/falcosecurity/deb stable main" | tee -a
/etc/apt/sources.list.d/falcosecurity.list apt-get update -y apt-get -y install
linux-headers-$(uname -r) apt-get install -y falco
```

Additional
https://www.youtube.com/watch?v=zgRFN3o7nJE&feature=emb_title
https://www.youtube.com/watch?v=8g-NUUmCeGI

Thank You

Reach out me on
Email: nilesh93.j@gmail.com
Linkedin: https://www.linkedin.com/in/nilesh93/