K8s Pen testing

Good Evening

Nidaa Saffarini
9/9/2020

K8s Pen testing

TALK Goals

1. k8s pertest Methodology

2. Difference of RISK , THREAT of web vulnerability in monolith app and container app
3. OWASP Attacks Vector in k8s cluster
4. main k8s security to do pentest in
5. Demo of a case study

K8s Black box Pen testing Process

K8s Pentest Methodology

Pentest K8S

1

Vulnerable Services and web application running in POD

1. Exploit app vulnerability
2. Take advantage from Privileged POD
3. Go inside Cluster

2

k8s components through exposed API endpoints

Scanning ports , Enumeration and Reconnaissance k8s components and network

Same Reasons

1. Priviliged POD
2. Mis configured RBAC
3.Mis Structured Network

K8s Pen testing Methodology 2

Attacking the Cluster Remotely

1. Subdomain Enumeration

2. Searching for Sensitive Information or Configuration Files in Github

3. Port scanning – External Port Visibility

Or Network Plugins

Port	Process
443/TCP	kube-apiserver
2379/TCP	etcd
6666/TCP	etcd
4194/TCP	cAdvisor
6443/TCP	kube-apiserver
8443/TCP	kube-apiserver
8080/TCP	kube-apiserver
10250/TCP	kubelet
10255/TCP	kubelet
10256/TCP	kube-proxy

K8s Pen testing Methodology 2

Attacking the Cluster Remotely

4. Checking Anonymous Access to the API Server

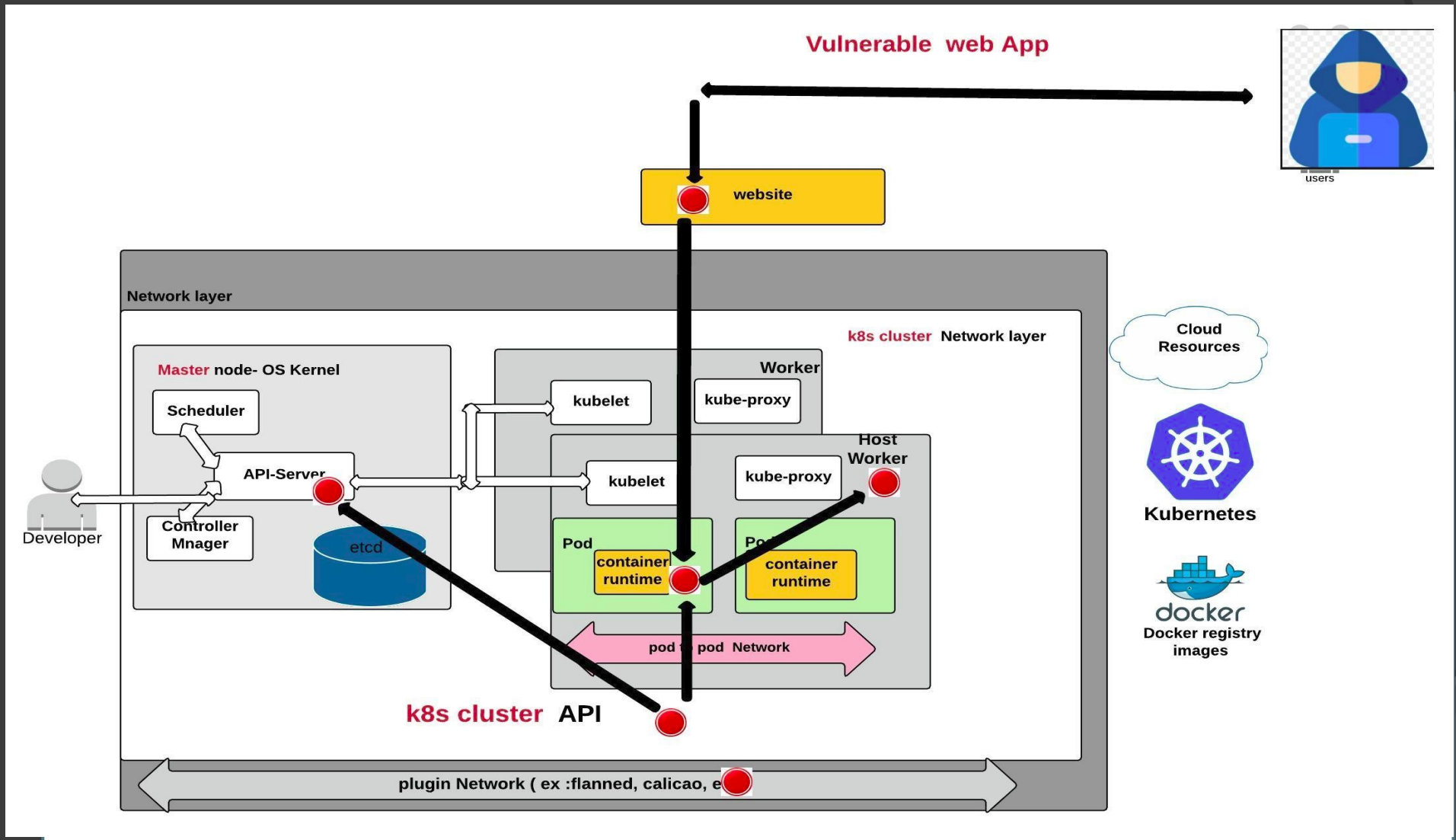
5. Checking for ETCD Anonymous Access

6. Checking Kubelet (Read Only Port) Information Exposure

curl commands

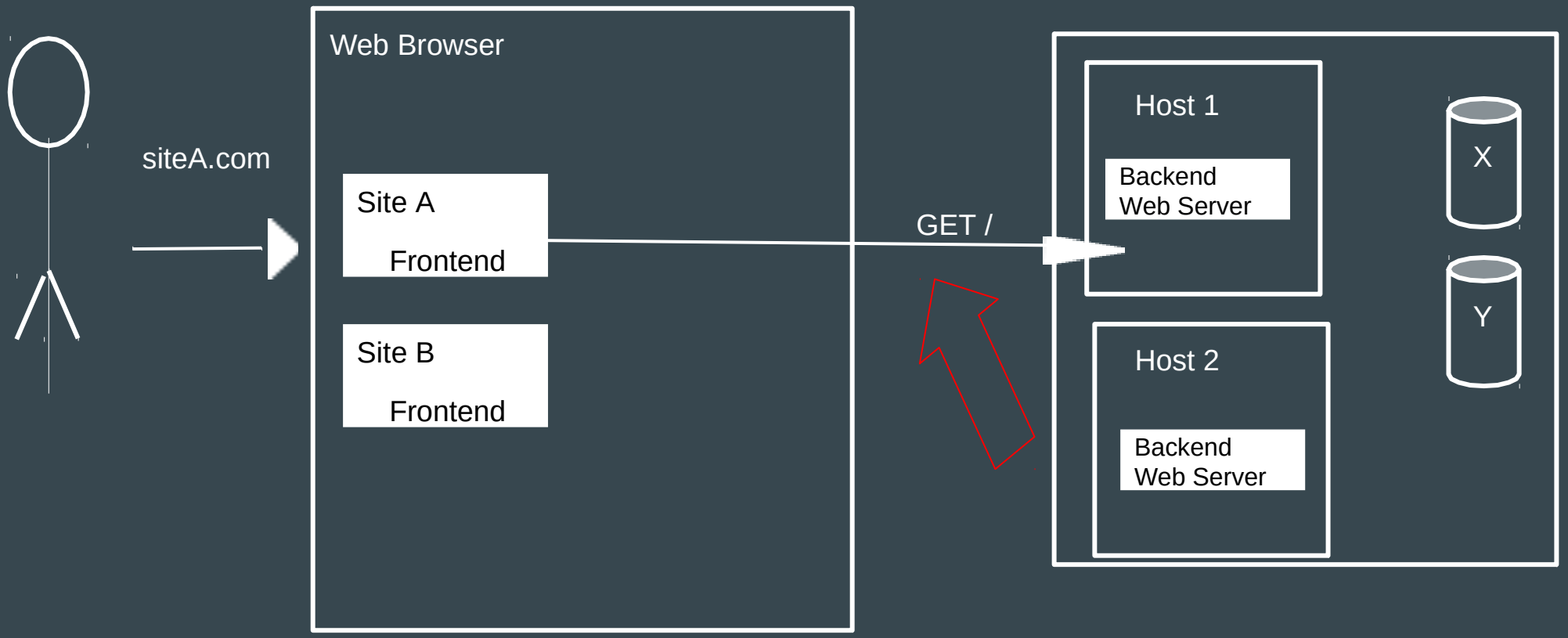
```
k8s API  : curl -K https://100.21.125.13:443
ETCD     : curl -K https://100.21.125.13:2379/version
Kubelete : curl -K https://100.21.125.13:10250
           curl -K https://100.21.125.13:10250/metrics
curl -K https://100.21.125.13:6443
curl -K https://100.21.125.13:6443/healthz
```

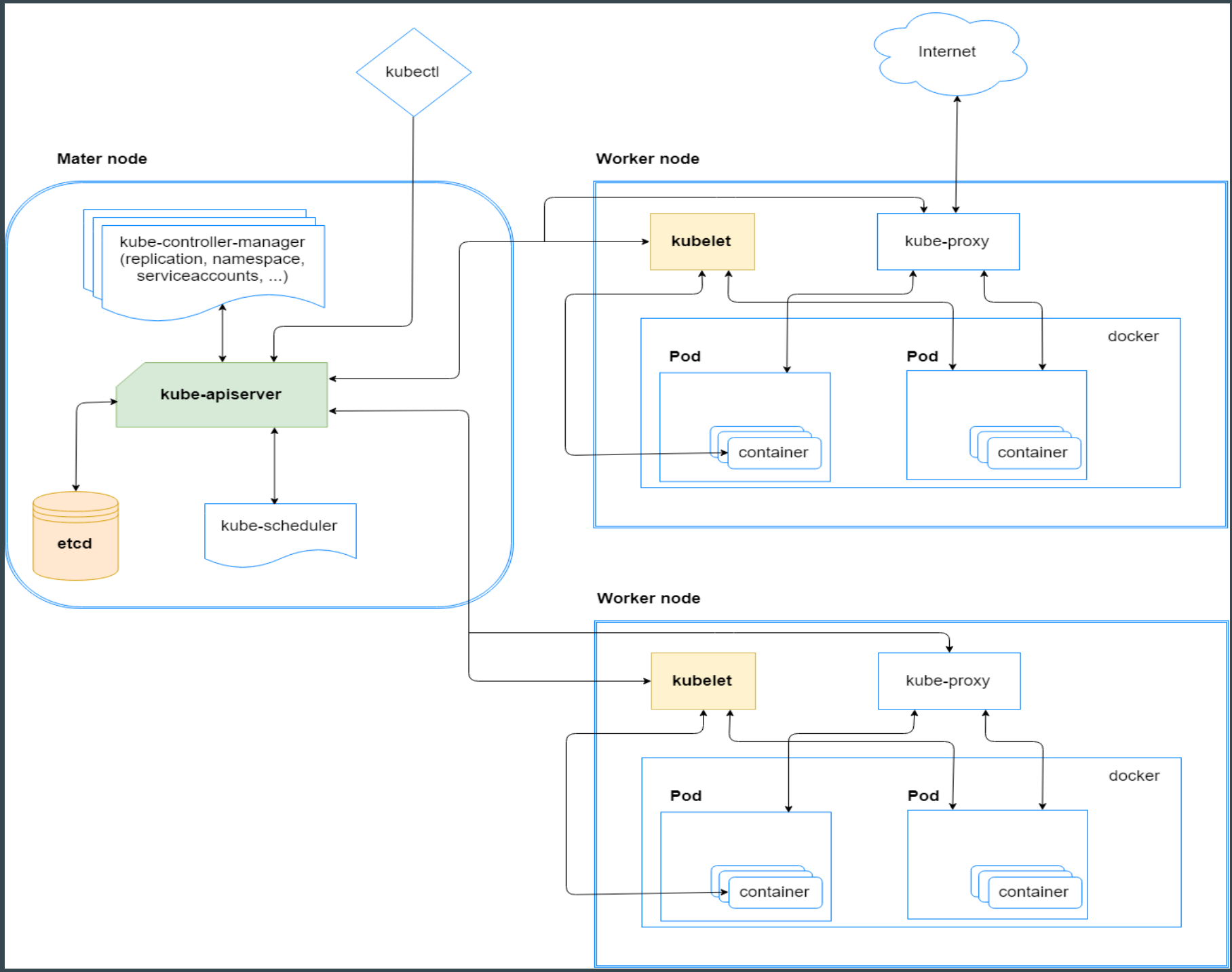
K8s Pen testing Methodology 1



What is WEB Vulnerabilities , RISK . Threat Before K8S

Security in monolith web application




kubectl
Internet
Mater node
Worker node
kube-controller-manager
(replication, namespace,
serviceaccounts, ...)
kubelet
kube-proxy
docker
Pod
Pod
kube-apiserver
container
container
etcd
kube-scheduler
Worker node
kubelet
kube-proxy
docker
Pod
Pod
container
container

OWASP k8s principle

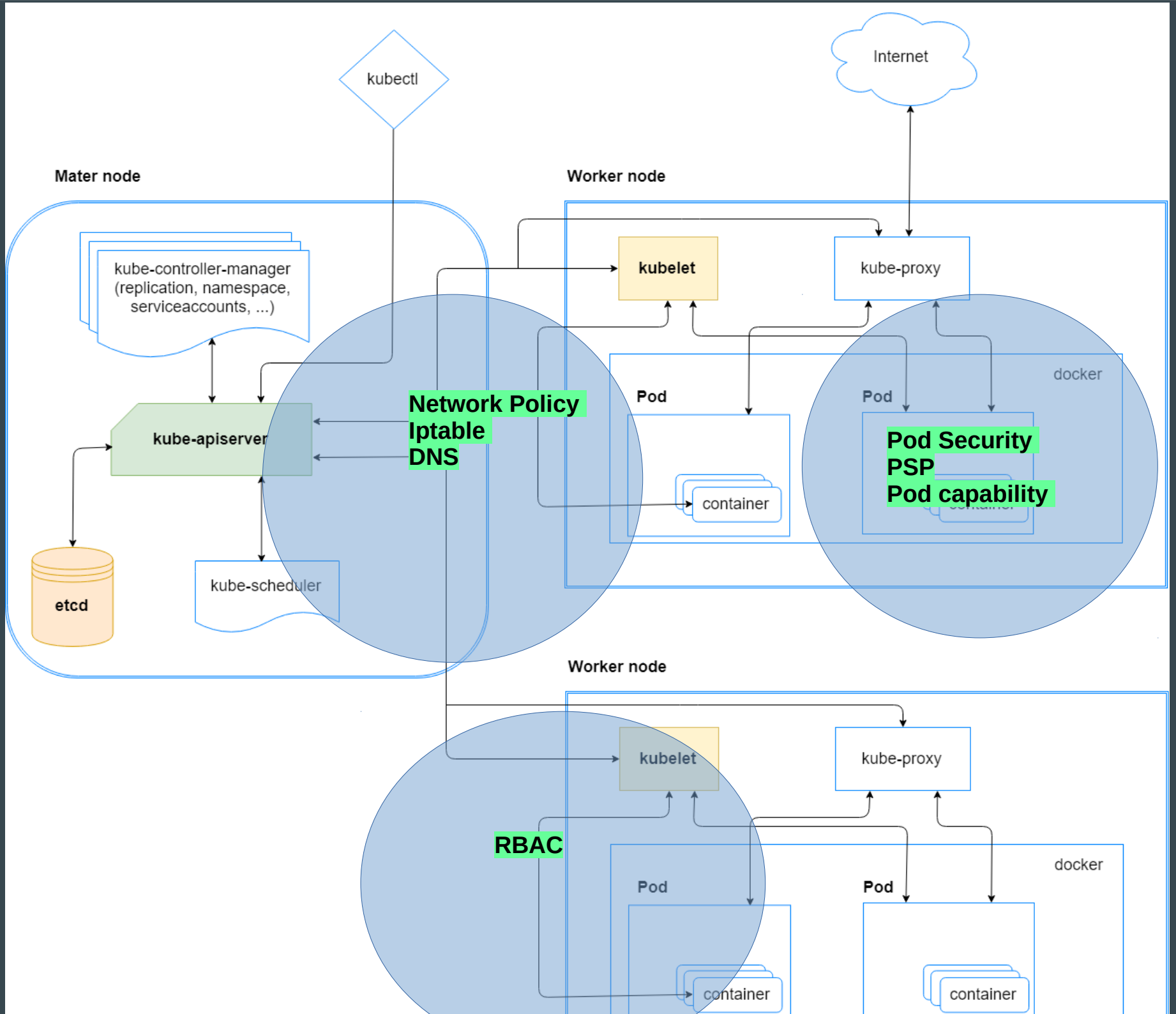
Kubernetes Security Testing Guide (KSTG)

The KSTG is (aims to be) a comprehensive manual for Kubernetes security analysts and red teamers. It aims to help DevSecOps Teams understand attacker TTPs and design effective countermeasures. KSTG propose to have the following high-level structure:

. Introduction to Kubernetes Architecture and its Components
. Kubernetes Cluster Threat Model
. Container Security Assessment
. Cluster Discovery and Recon
. Cluster Security Assessment
. Auditing against CIS Benchmarks

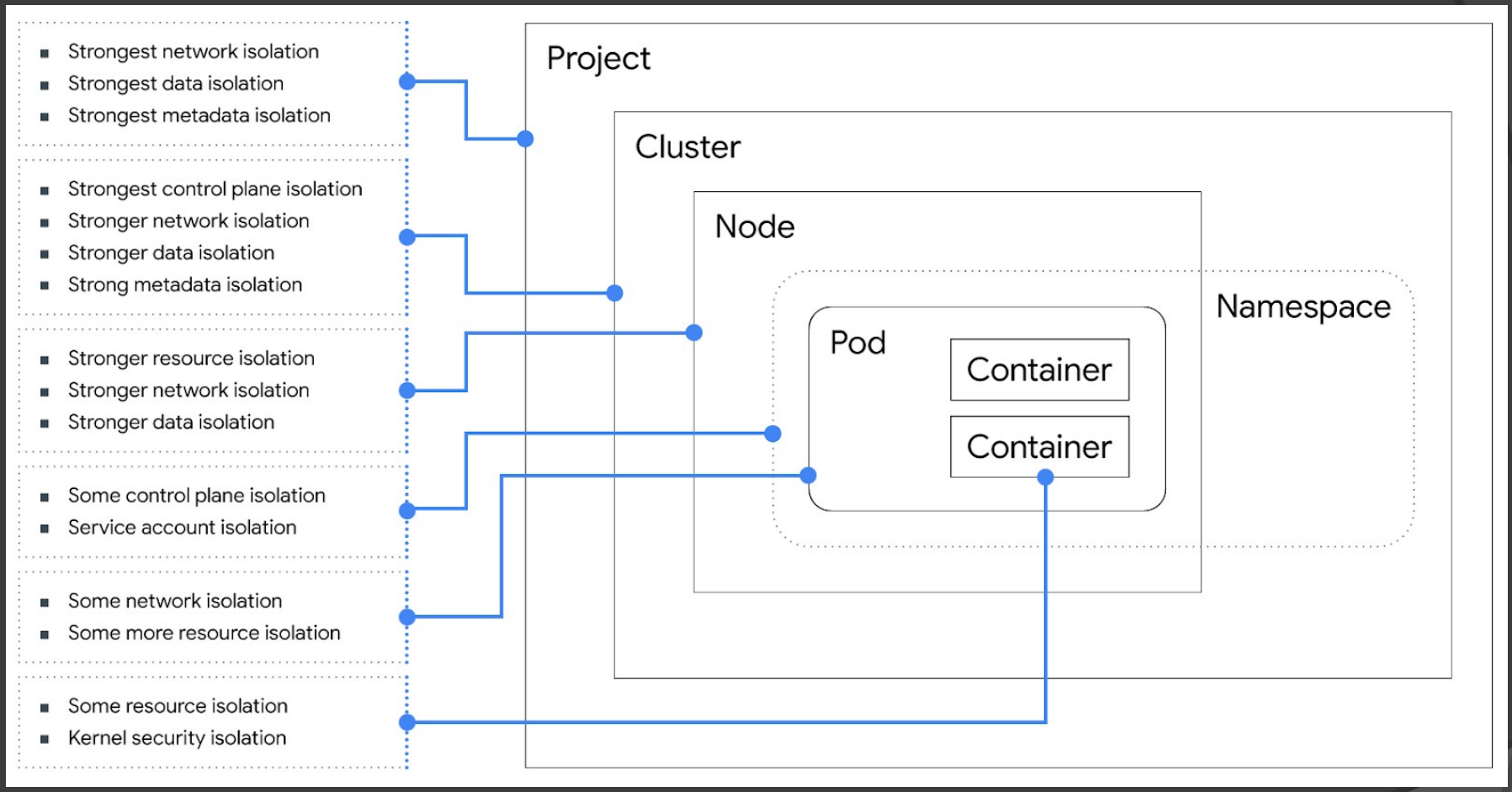
OPEN WEB APPLICATION SECURITY PROJECT –OWASP

	OWASP TOP 10 VULNERABILITIES	Risk and threat	Risk and threat
	2020	Monolith web application	Microservices web application in k8s Cluster
A1	**Injection**	DB	DB
A2	**Broken Authentication**	Web app	App and Pod capability Cluster API
A3	**Sensitive data exposure**	Web server , host	K8s => , services , secrets, configmap ...etc
A4	**XML External Entities (XXE)**		k8s => Pod cabability ,RBAC
A5	**Broken Access Control**	Host, Network	K8s => Broken RBAC Hosts , Networks
A6	**Security Misconfiguration**	Iptables , firewalls	Rbac, network policy , pod security , helm ...etc
A7	**Cross Site Scripting**	Frontend app	
A8	**Insecure Deserialization**		
A9	**Using Components with Known Vulnerabilities**	Web server , library code	Library code in docker, k8s cluser component
A10	**Insufficient Logging and**	Web server	Cluster component


kubectl
Internet
Mater node
Worker node
kube-controller-manager
(replication, namespace,
serviceaccounts, ...)
kubelet
kube-proxy
Network Policy
Iptable
DNS
kube-apiserver
Pod
docker
Pod
Pod Security
PSP
Pod capability
container
etcd
kube-scheduler
Worker node
kubelet
kube-proxy
RBAC
docker
Pod
Pod
container
container

K8s and CIA

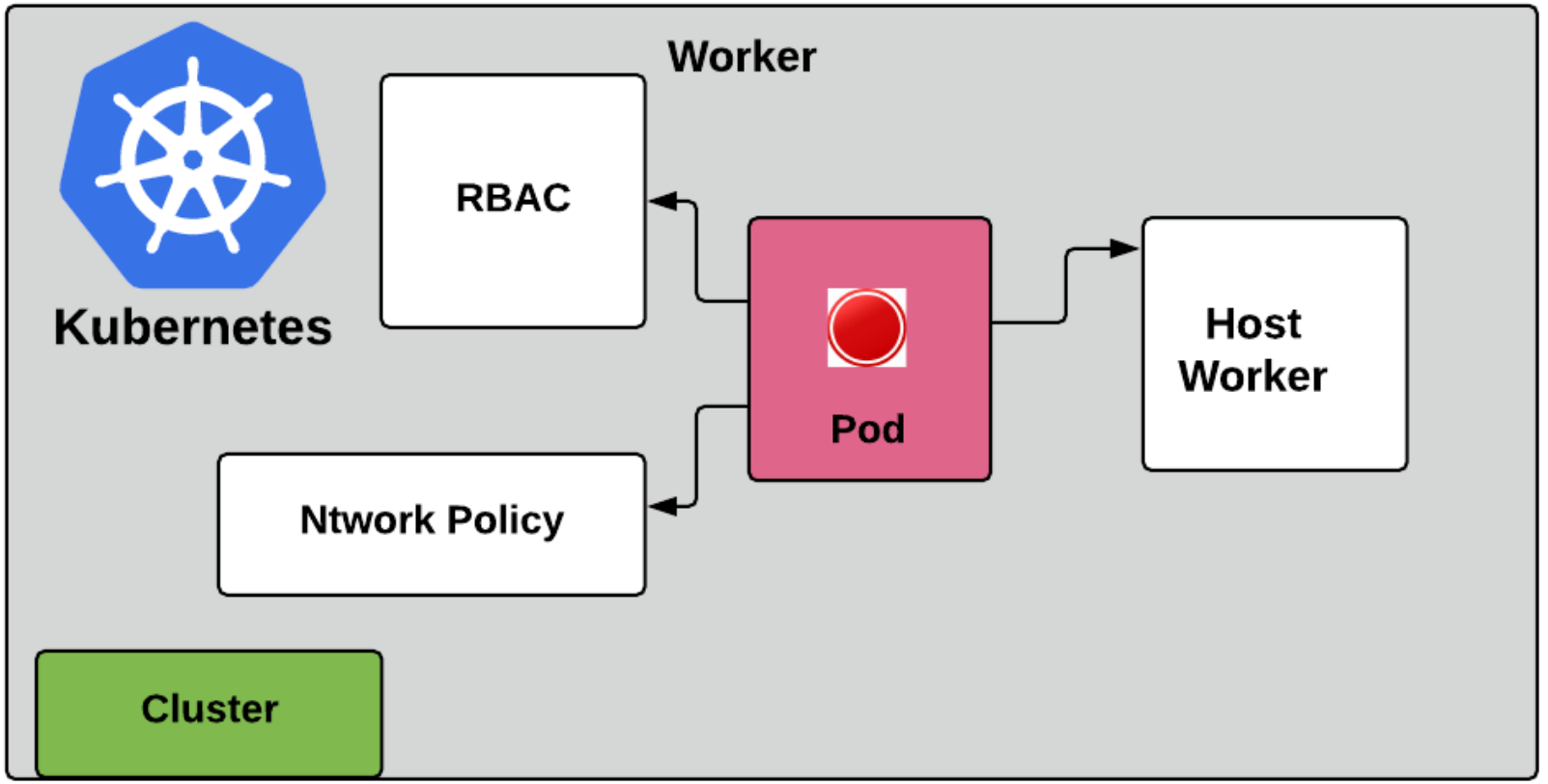
- Confidentiality
  - Access Control
    - Hardware
    - Firewalls
  - System Isolation
    - Different levels
    - Zones
- Integrity
  - Hardware
  - Software
- Availability :
  - Scalability
  - High Availability
  - Reliability

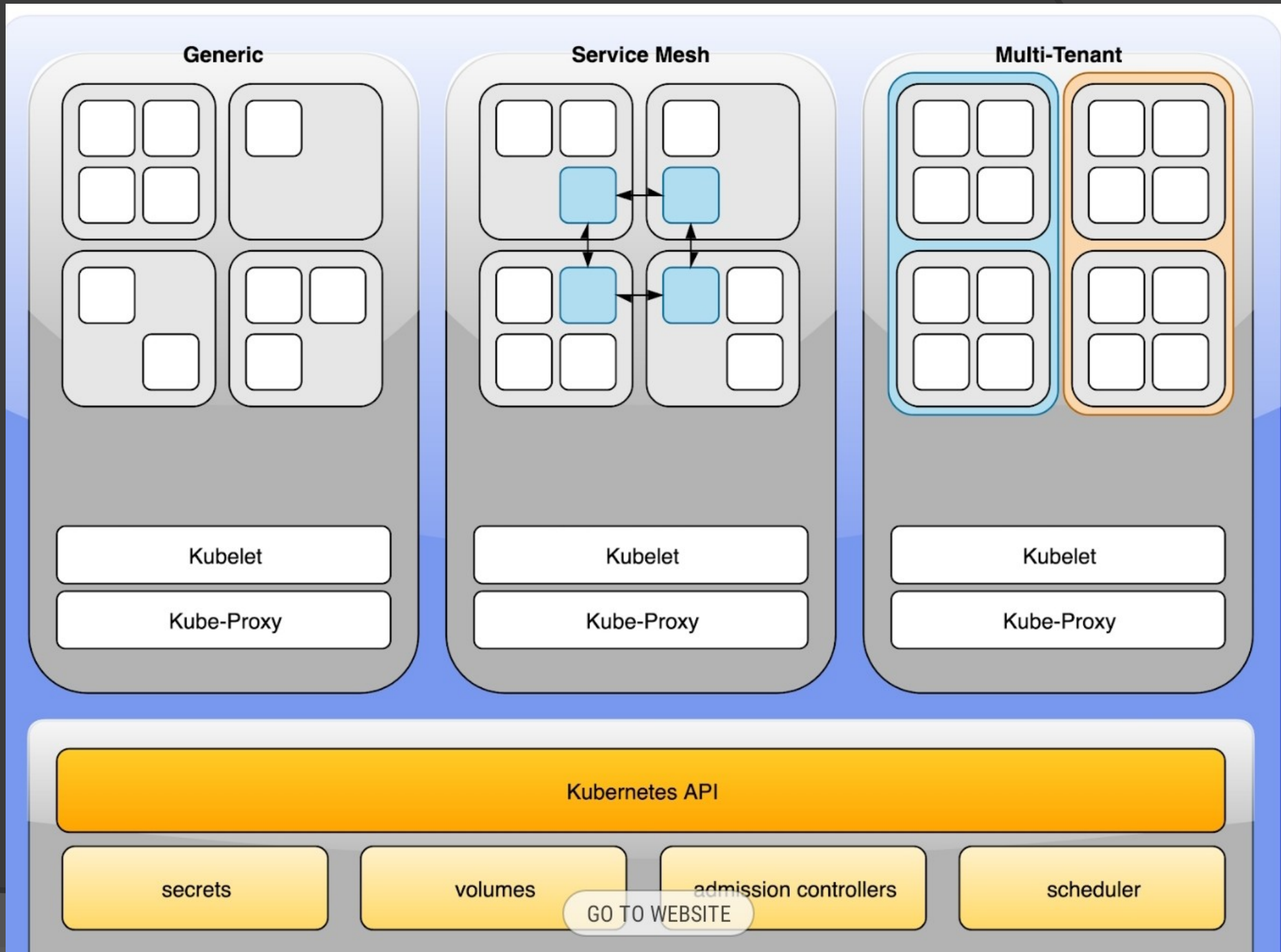

Strongest network isolation
Strongest data isolation
Strongest metadata isolation
Strongest control plane isolation
Stronger network isolation
Stronger data isolation
Strong metadata isolation
Stronger resource isolation
Stronger network isolation
Stronger data isolation
Some control plane isolation
Service account isolation
Some network isolation
Some more resource isolation
Some resource isolation
Kernel security isolation
Project
Cluster
Node
Namespace
Pod
Container
Container

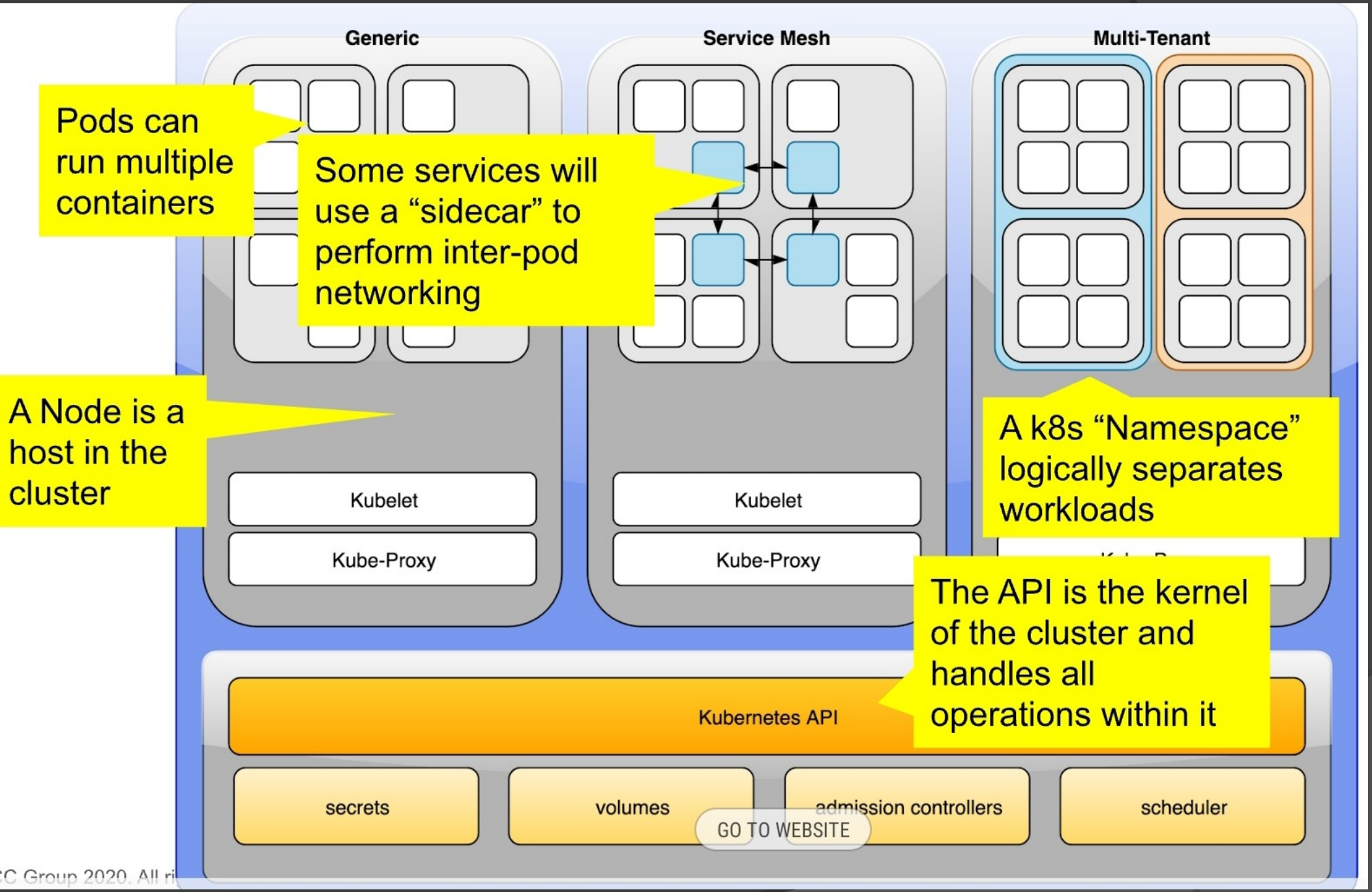
k8s attacks vector

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	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	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)		Access cloud resources	Connect from Proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files	
Exposed Dashboard	SSH server running inside container					Instance Metadata API	Writable volume mounts on the host	
							Access Kubernetes dashboard	
							Access tiller endpoint	

Why pod is important

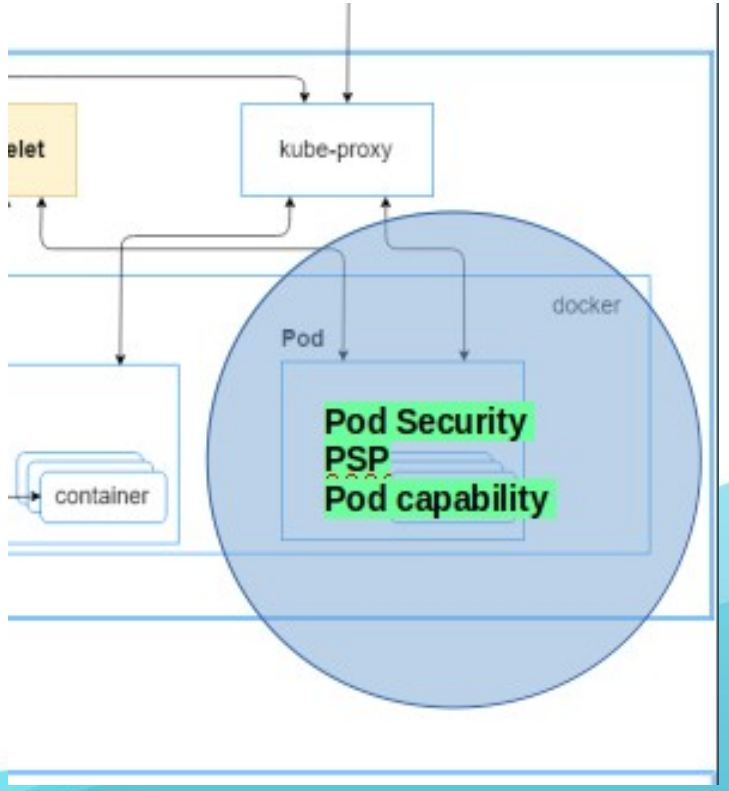



Generic
Service Mesh
Multi-Tenant
Kubelet
Kube-Proxy
Kubelet
Kube-Proxy
Kubelet
Kube-Proxy
Kubernetes API
secrets
volumes
admission controllers
scheduler
GO TO WEBSITE



CC Group 2020. All ri

CONTAINERS AND PODS



aqua

WHAT ARE CONTAINERS?

Way of isolating and restricting Linux processes

- **Isolation**
  - **Namespaces**
- **Capabilities**
- **Restriction**
  - **Cgroups**
  - **SecComp**

LINUX NAMESPACES

Namespace	Constant	Isolates
Cgroup	**CLONE_NEWCGROUP**	Cgroup root directory
IPC	**CLONE_NEWIPC**	System V IPC, POSIX message queues
Network	**CLONE_NEWNET**	Network devices, stacks, ports, etc.
Mount	**CLONE_NEWNS**	Mount points
PID	**CLONE_NEWPID**	Process IDs
User	**CLONE_NEWUSER**	User and group IDs
UTS	**CLONE_NEWUTS**	Hostname and NIS domain name
TIME	**CLONE_TIME**	Time, coming soon???
SYSTEMD	**CLONE_SYSTEMD**	systemd in a namespace, who ordered that?

KERNEL CAPABILITIES

```
CAP_AUDIT_CONTROL,  CAP_AUDIT_READ,  CAP_AUDIT_WRITE,  CAP_BLOCK_SUSPEND,
CAP_CHOWN,CAP_DAC_OVERRIDE, CAP_DAC_READ_SEARCH, CAP_FOWNER,  CAP_FSETID,
CAP_IPC_LOCK, CAP_IPC_OWNER, CAP_KILL, CAP_LEASE, CAP_LINUX_IMMUTABLE,
CAP_MAC_ADMIN,CAP_MAC_OVERRIDE, CAP_MKNOD,  CAP_NET_ADMIN ,
CAP_NET_BIND_SERVICE, CAP_NET_BROADCAST,  CAP_NET_RAW , CAP_SETGID,  CAP_SETFCAP,
CAP_SETPCAP, CAP_SETUID,                CAP_SYS_ADMIN , CAP_SYS_BOOT,
CAP_SYS_CHROOT, CAP_SYS_MODULE, CAP_SYS_NICE, CAP_SYS_PACCT,  CAP_SYS_PTRACE,
CAP_SYS_RAWIO, CAP_SYS_RESOURCE,     CAP_SYS_TIME ,  CAP_SYS_TTY_CONFIG,
CAP_SYSLOG, CAP_WAKE_ALARM, CAP_INIT_EFF_SET
```

38 Flags in a 64 Bit array

Privileged pods

- **When do we need privileged pods?**
  - Let containers use **host's resources**
  - eg. manipulate the network stack / access graphic card
- **Security risk**
  - Process in privileged containers == root process on the host
  - An attacker can basically do ANYTHING....

Unfortunately, we have to look into the container

```
cat /proc/1/status -- NoNewPrivs:    1
```

In Linux, the `execve` *system call can grant more privileges to a newly-created process than its parent process. Considering security issues, since Linux kernel v3.5, there is a new flag named* `no_new_privs` *added to prevent those new privileges from being granted to the processes.*

this is a stupid idea

By default, ie when `allowPrivilegeEscalation=nil`, we will set `no_new_privs=true` with the following exceptions:

- when a container is `privileged`
- when `CAP_SYS_ADMIN` is added to a container
- when a container is not run as root, uid `0` (to prevent breaking suid binaries)

WHAT ARE KUBERNETES PODS?

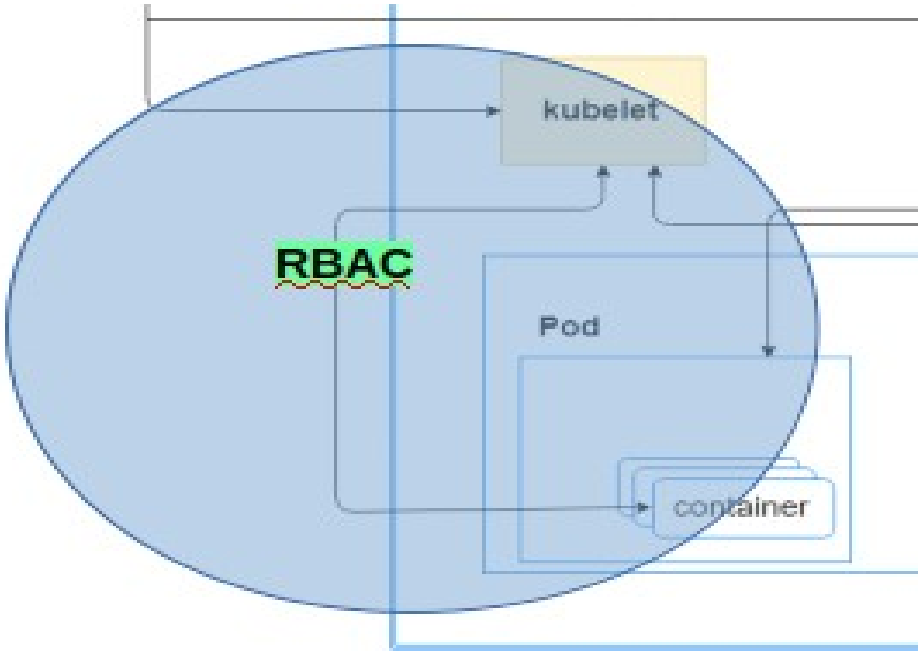
- Core Concept the Kubernetes Microservice
- Bunch of Containers with the same
  - Lifecycle: live together, die together
  - Network:
    - same ip address,
      same 127.0.0.0/8
    - same routes
    - same iptables
    - same DNS
  - Volumes: can share data
  - One common task
  - Init Tasks
  - Live and Readiness
    Checks

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
      containe
  rs:
  - name: nginx
    image: nginx
```

DETECT PRIVILEGES

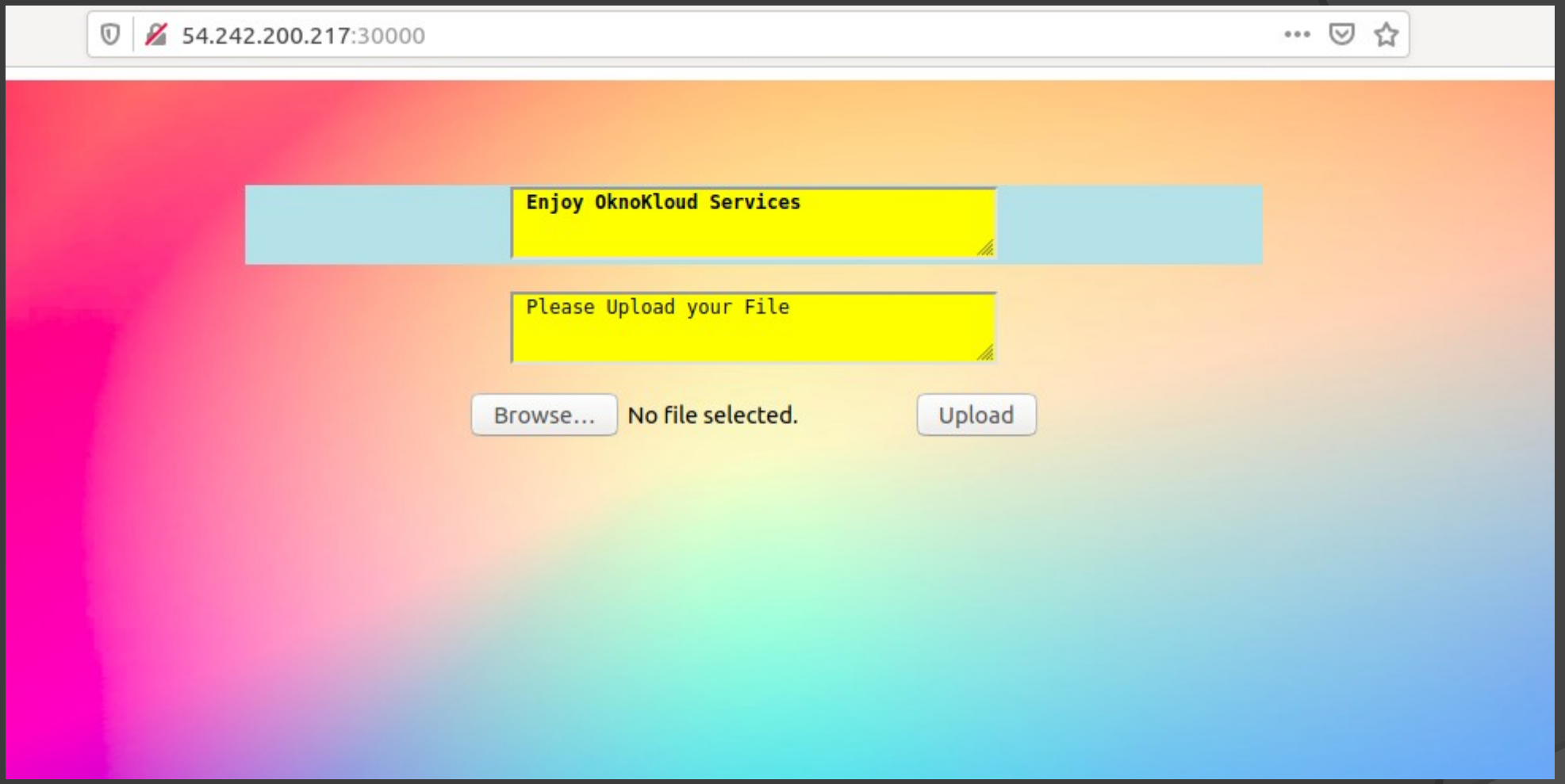
```
kubectl get pods --all-namespaces -o jsonpath='
{range .items[*]}
{range .spec.initContainers[*]}
{.image}{"\t"}
{.securityContext}
{.end}{"\n"}
{end}
' | sort |uniq
```

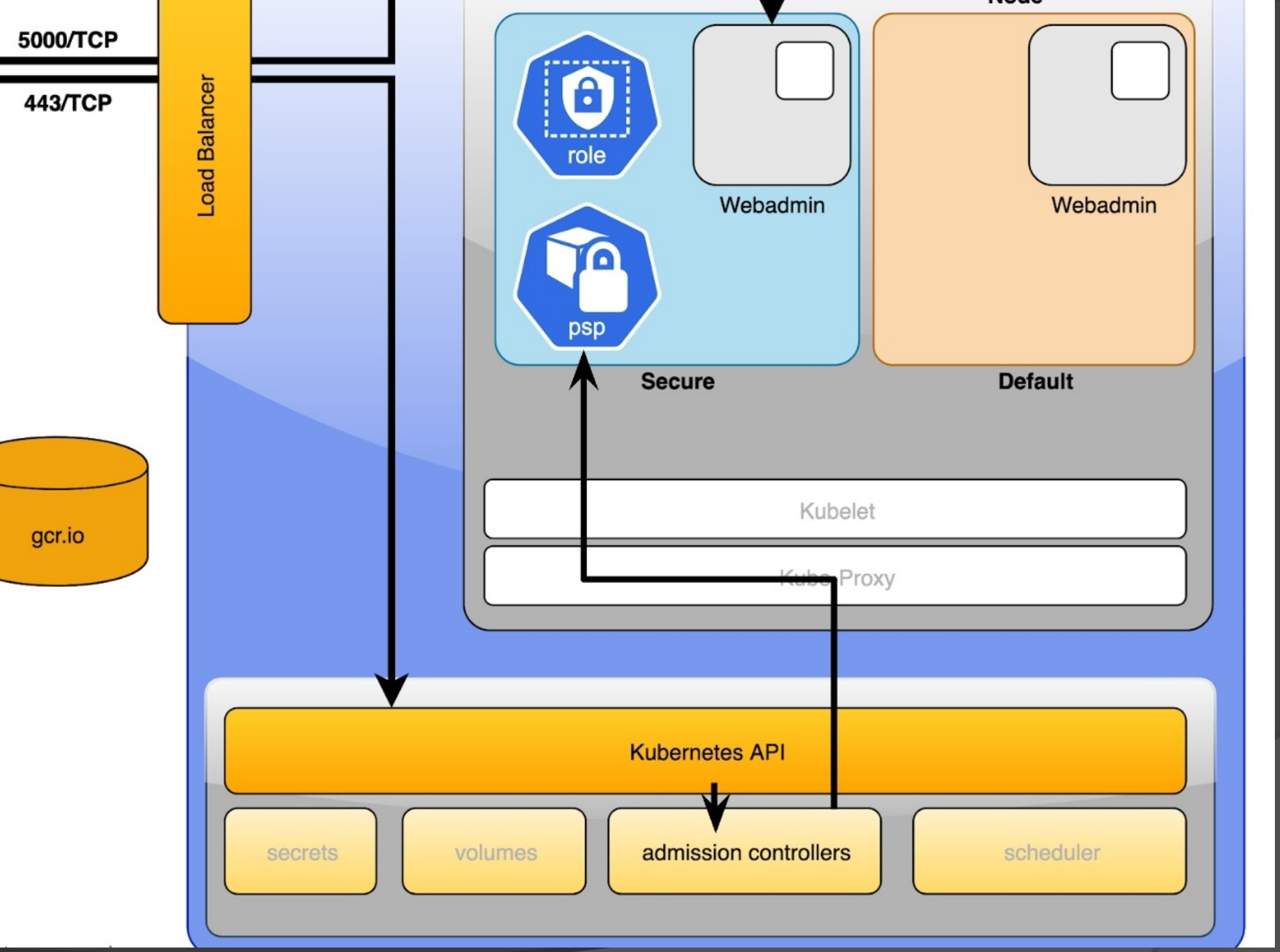
RBAC Security

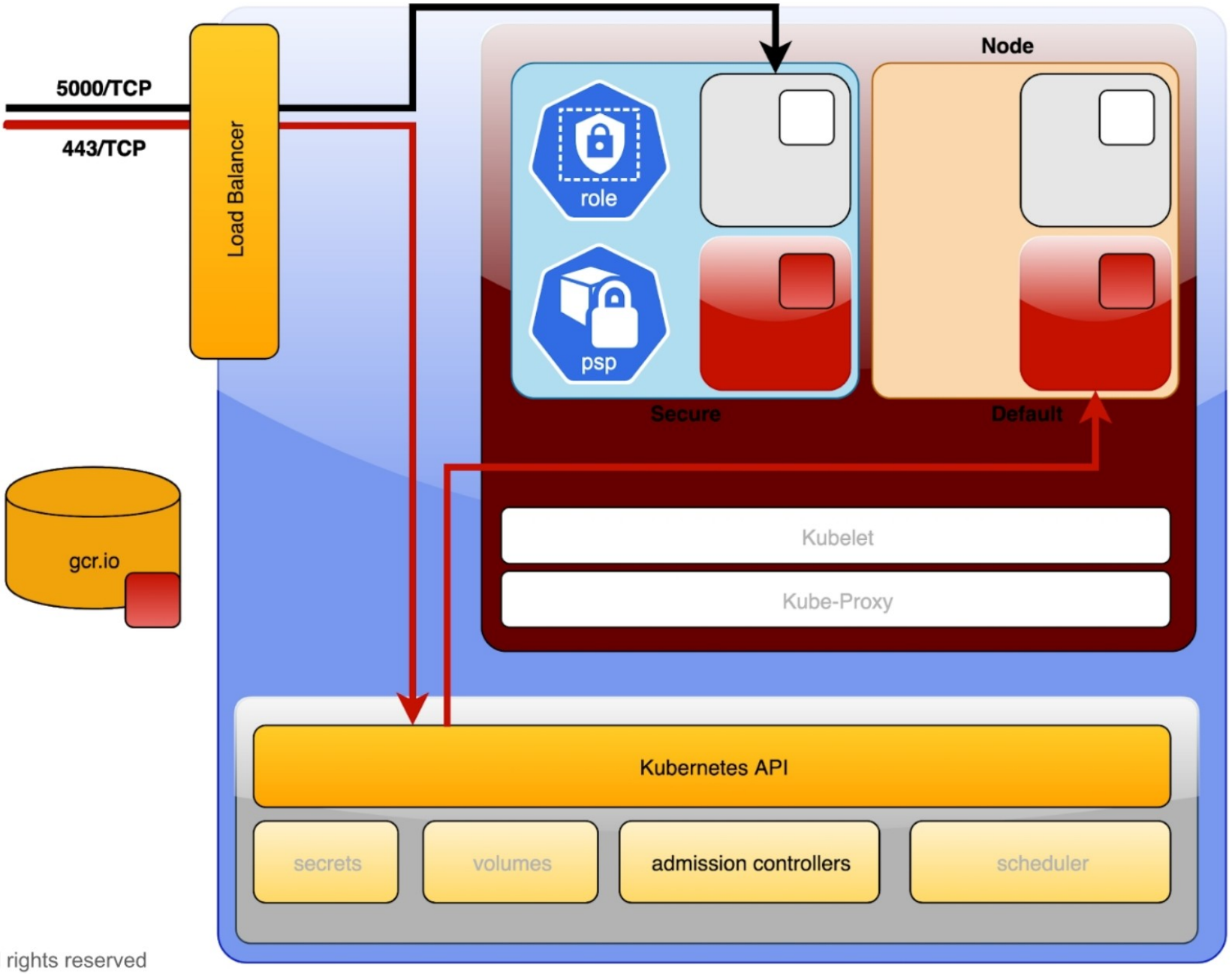


aqua

- Case Study : Oknokloud Company

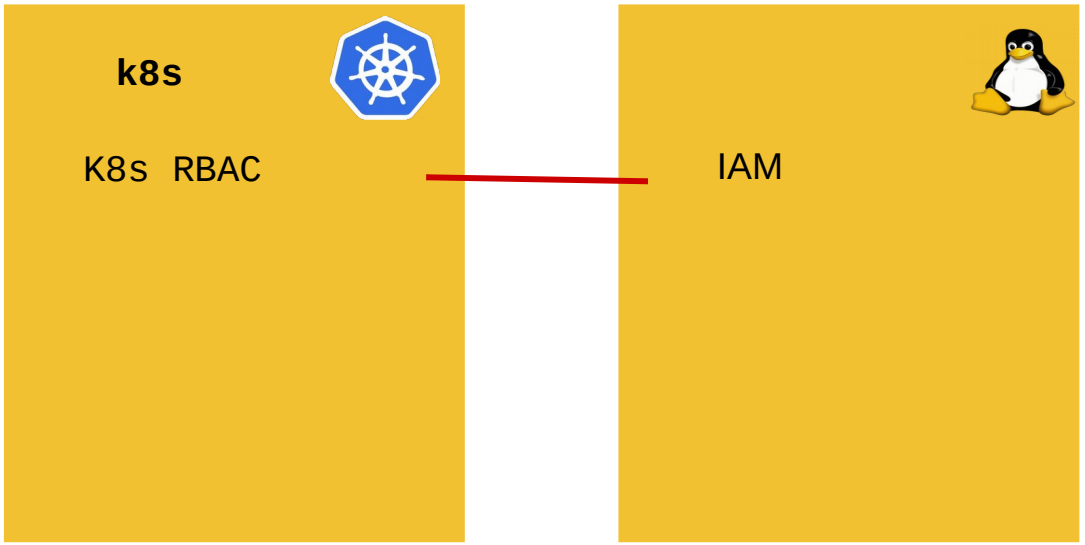

54.242.200.217:30000
Enjoy OknoKloud Services
Please Upload your File
Browse...
No file selected.
Upload

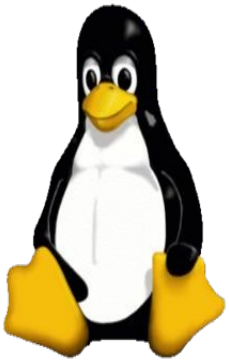

5000/TCP
443/TCP
Load Balancer
role
psp
Webadmin
Webadmin
Secure
Default
gcr.io
Kubelet
Kube Proxy
Kubernetes API
secrets
volumes
admission controllers
scheduler



NCC Group 2020. All rights reserved

RBAC security

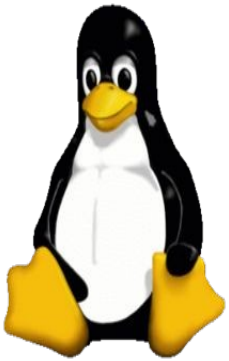




IN Linux Everything is File

IN Kubernetes Everything is Resource

Files have owners and permissions



permissions

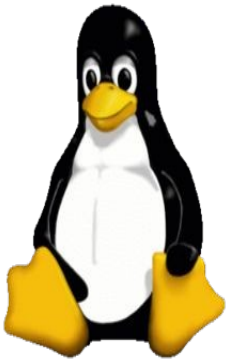
owner group

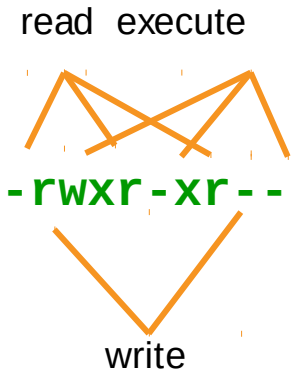
owner everyone

```
-rwxr-xr--  1 nidaa    ubuntu   6340   Aug  3 00:58 myapp
```

group

Files have owners and permissions





owner group

```
-rwxr-xr--   1 nidaa    ubuntu   6340   Aug  3 00:58 myapp
```

Read
Anyone can read
myapp

Write
Only user nidaa can write
myapp

Execute
User nidaa can run
myapp

Any user in group *staff* can run
myapp

Linux		Kubernetes 
File		Resources
File owner		?
		?

Linux

File

File owner

File permissions

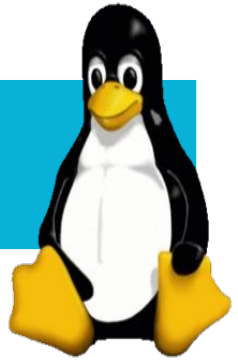
Kubernetes

Resources

Resources don’t have owners

Resources don’t have permissions

Linux



File

Example

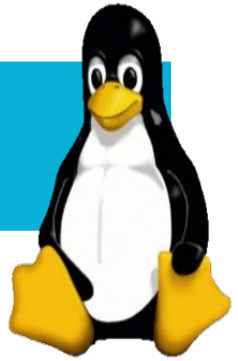
Running *myapp*

Kubernetes

Resources

Creating a pod to run *myapp*

Linux



read

execute
write

Kubernetes

get list
watch
create
delete patch
 update
use, bind, ...

verbs

```
-rwxr-xr--    1 nidaa   staff  9567 Mar 08:22 myapp
```

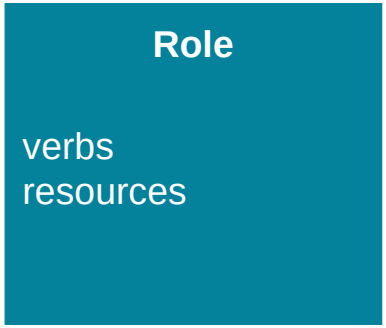
<verbs> you can do to a file called <name>

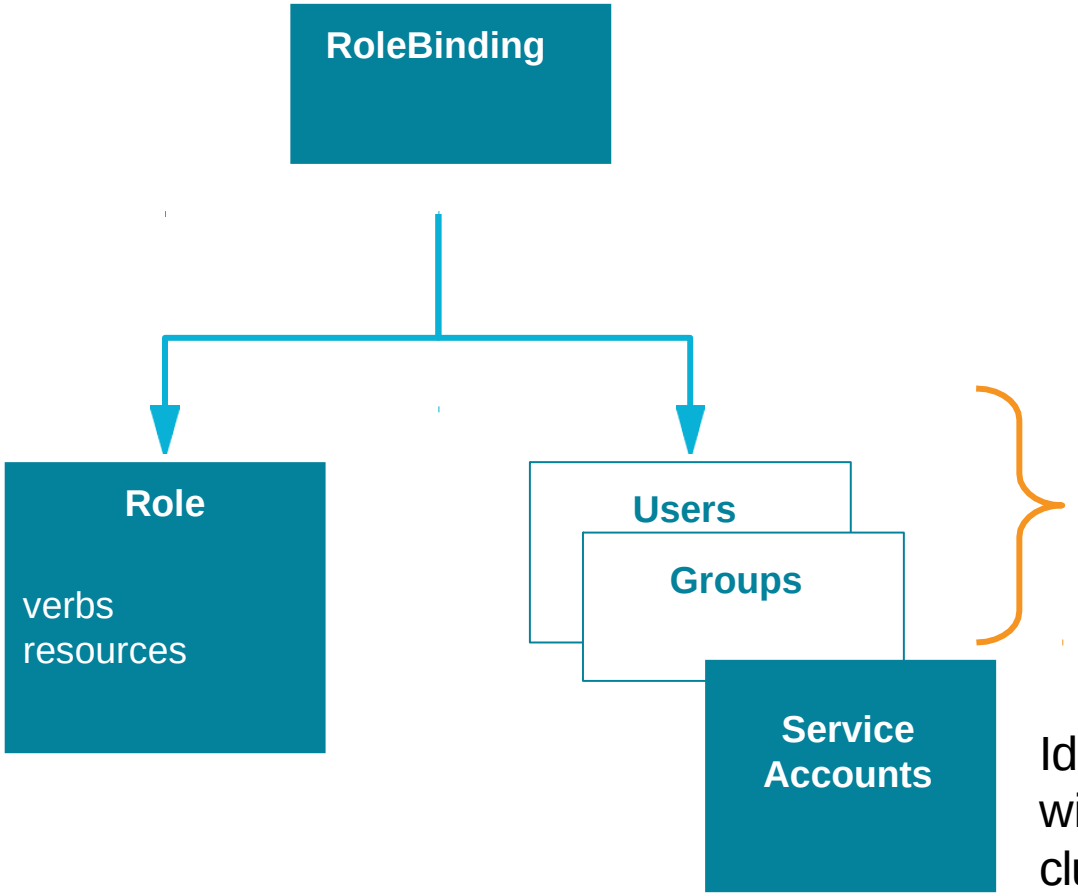
Role

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: my-role  namespace: my-project
rules:
- apiGroups:
  - ""
  Resources:

  - pods
  verbs:
  - create
  - get
  - list
```

<verbs> you can do to <resources>



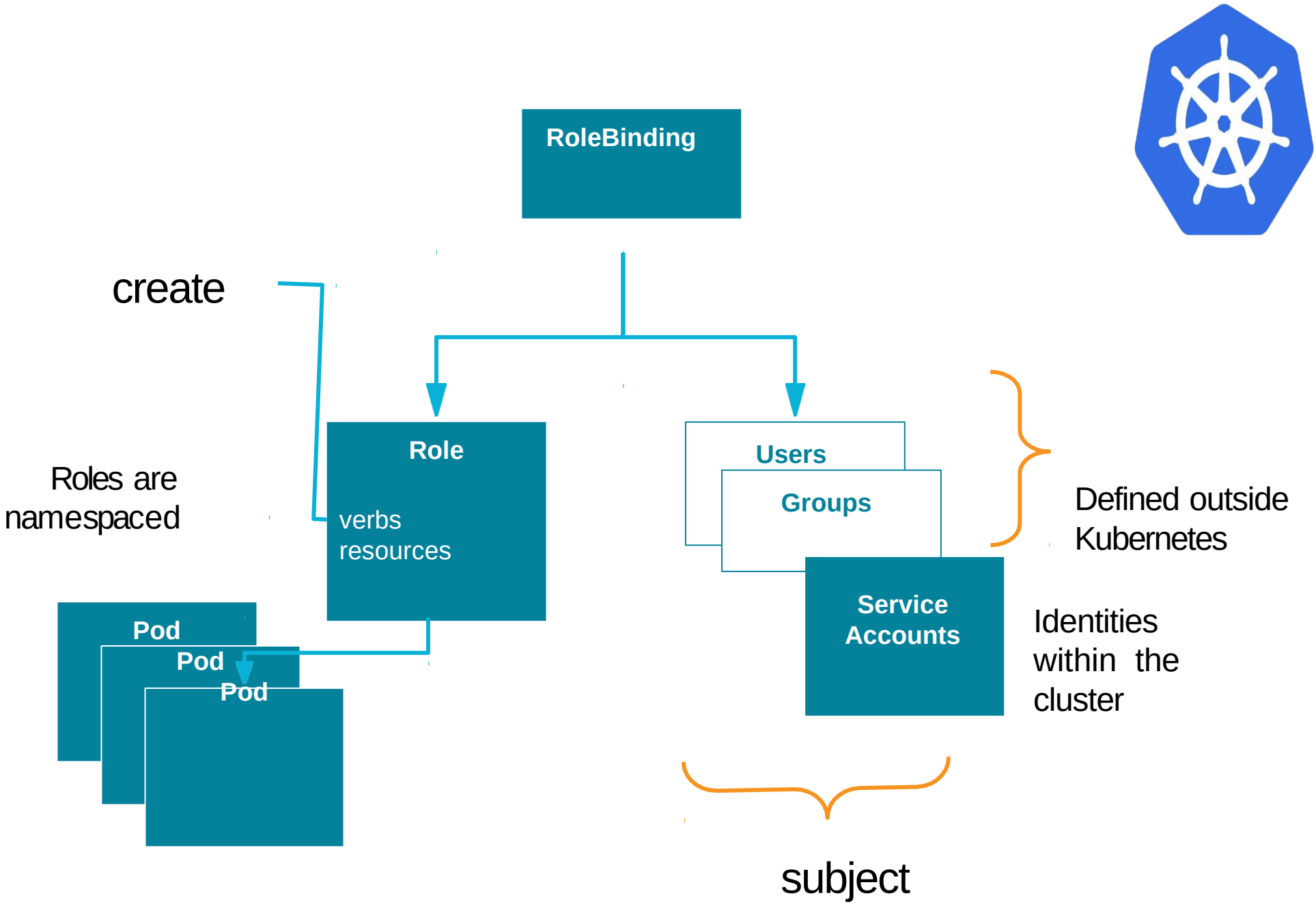


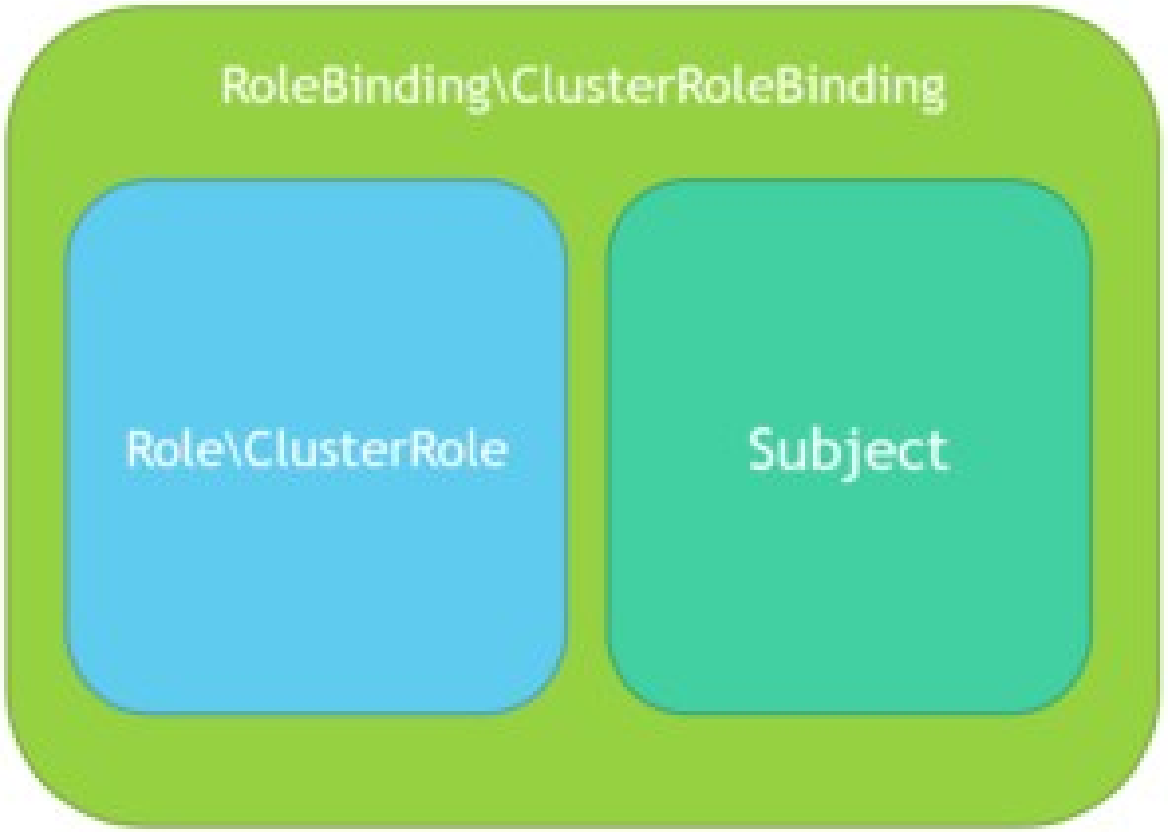
Roles are namespaced

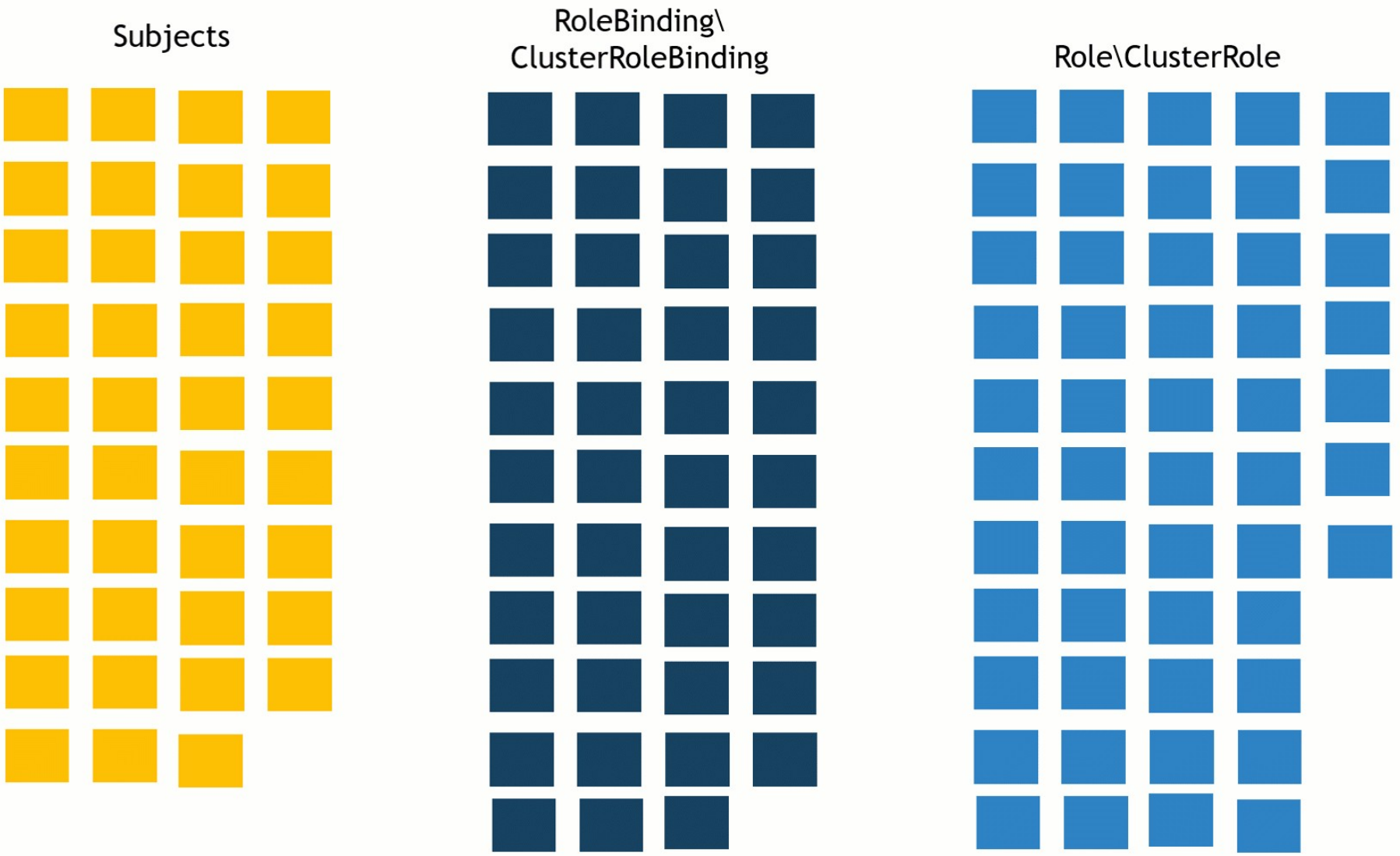
Defined outside Kubernetes

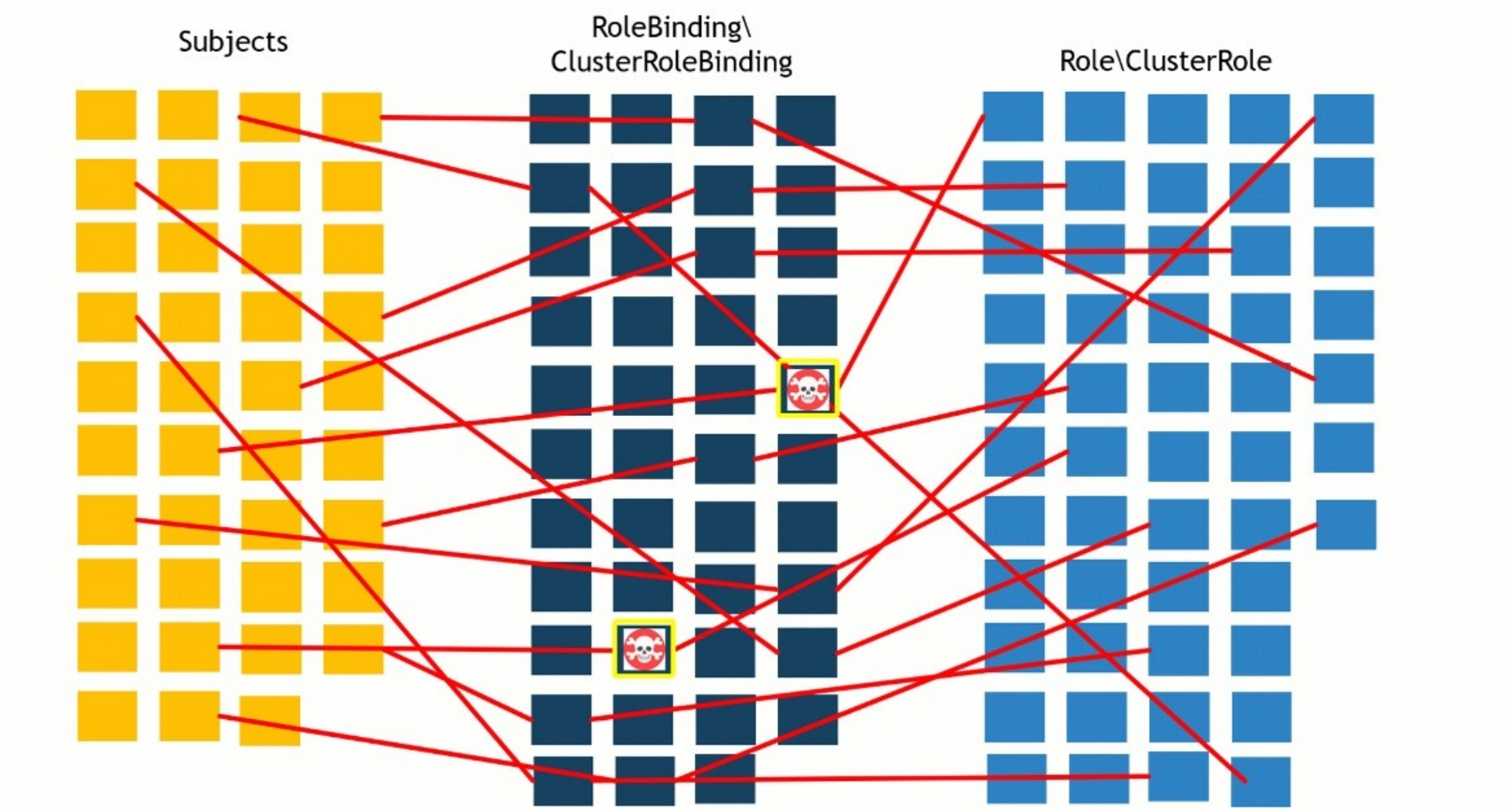
Identities within the cluster

aqua


RoleBinding
create
Role
verbs
resources
Roles are
namespaced
Pod
Pod
Pod
Users
Groups
Service
Accounts
Defined outside
Kubernetes
Identities
within the
cluster
subject


RoleBinding\ClusterRoleBinding
Role\ClusterRole
Subject

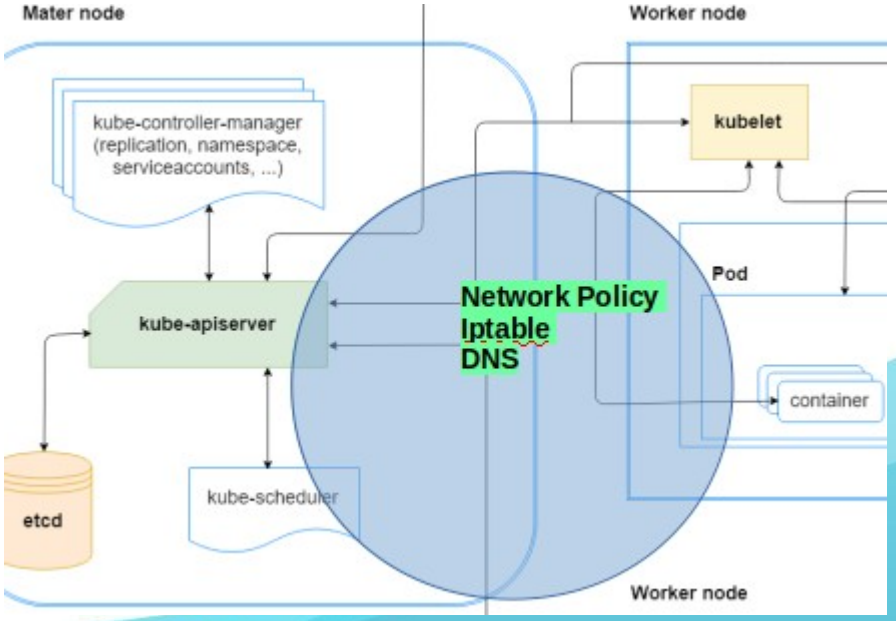

Subjects
RoleBinding\
ClusterRoleBinding
Role\ClusterRole


Subjects
RoleBinding\
ClusterRoleBinding
Role\ClusterRole

Can I create a pod?

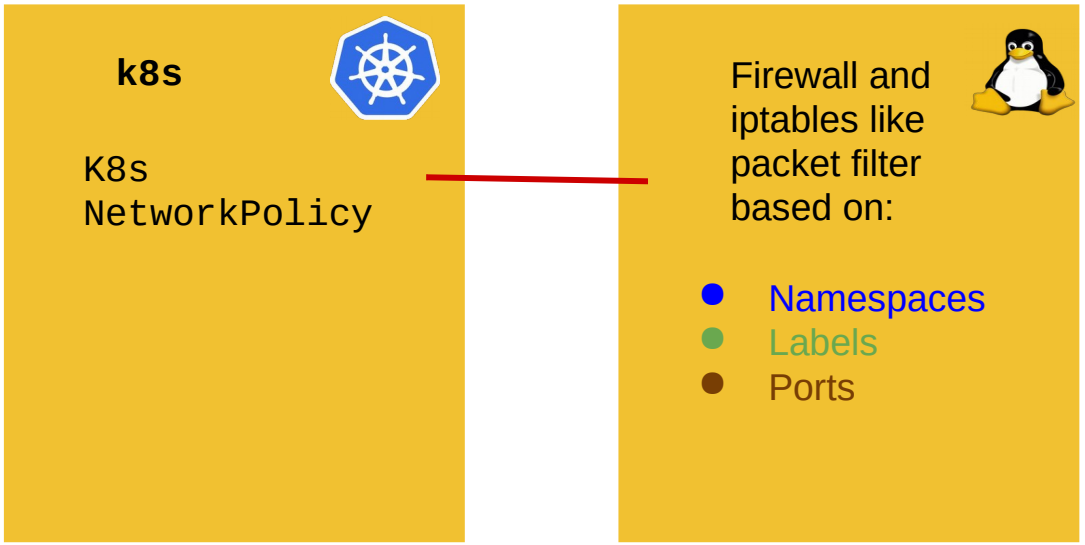
```
$ kubectl auth can-i create pods
$ kubectl auth can-i list pods
```

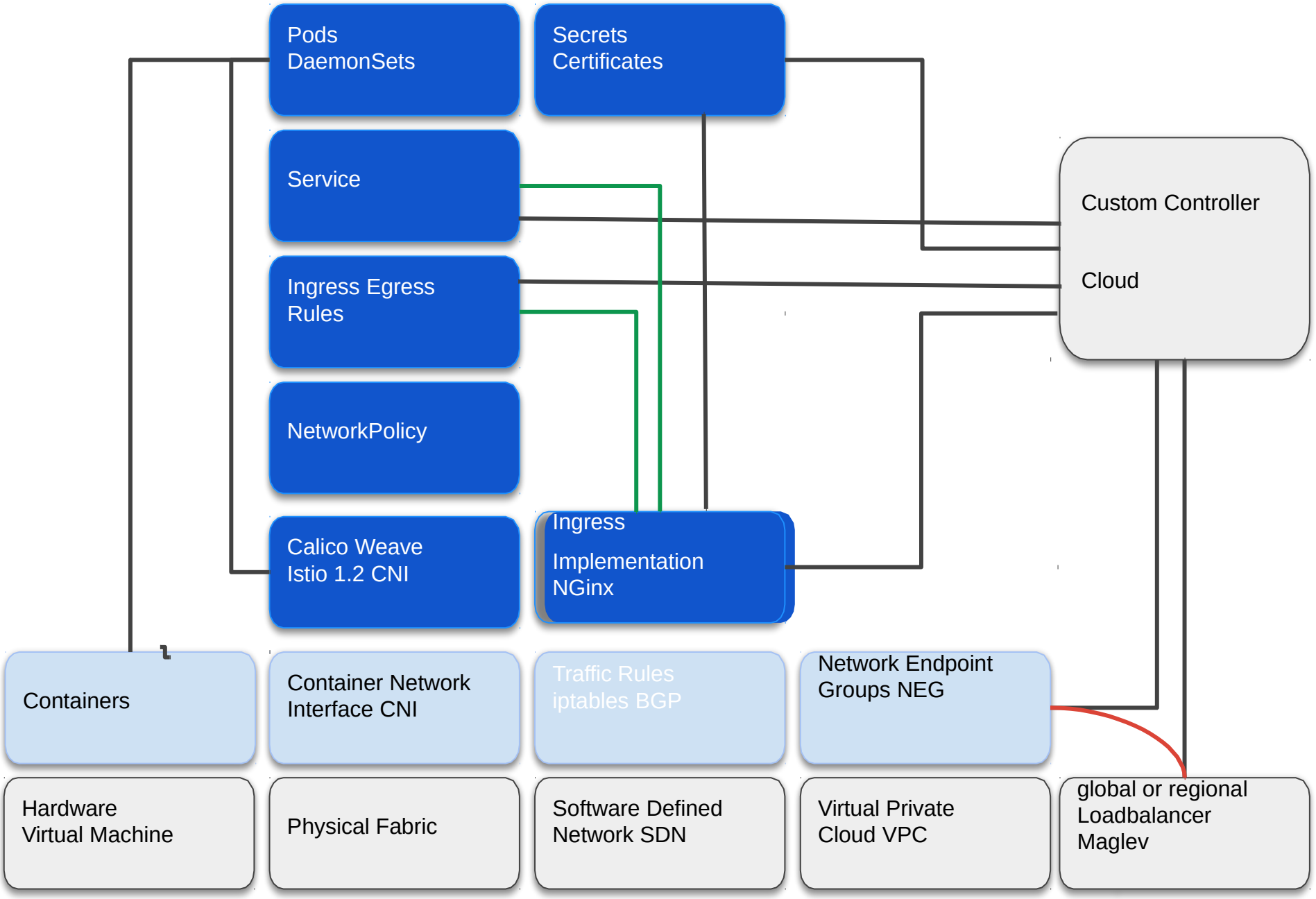
NETWORK Security



aqua

network security




Pods
DaemonSets
Secrets
Certificates
Service
Custom Controller
Ingress Egress
Rules
Cloud
NetworkPolicy
Calico Weave
Istio 1.2 CNI
Ingress
Implementation
NGinx
Containers
Container Network
Interface CNI
Traffic Rules
iptables BGP
Network Endpoint
Groups NEG
Hardware
Virtual Machine
Physical Fabric
Software Defined
Network SDN
Virtual Private
Cloud VPC
global or regional
Loadbalancer
Maglev

- **Network policy guidelines**

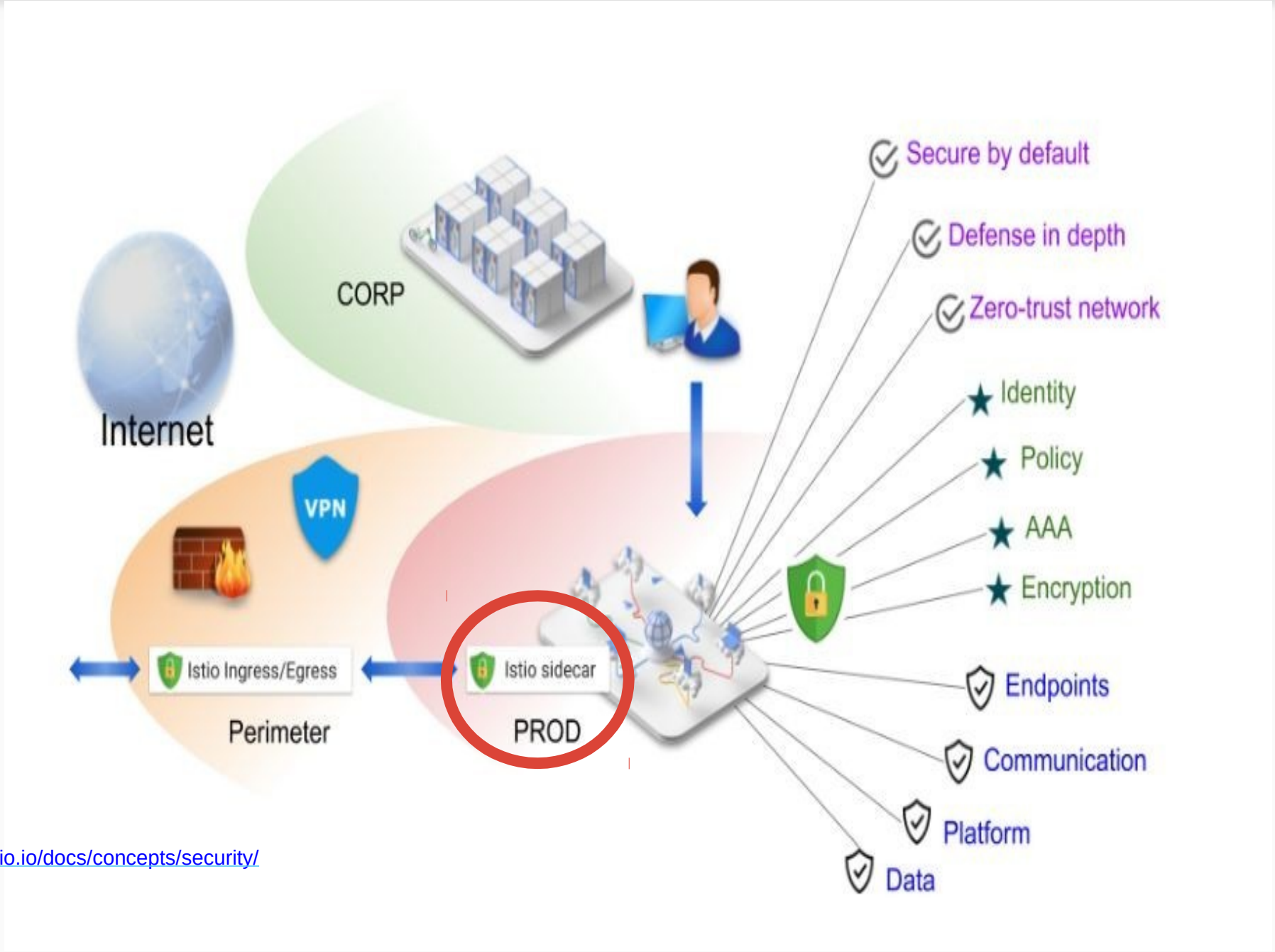
- Label your workloads properly
- Isolate workloads from each other
- Restrict income traffic to the kube-system (except kube-dns)
- Consider limit egress to the internet

NETWORK

- Calico
- NetworkPolicy
- Ingress

- Solution : Zero Trust (Istio) ??

WE USE ISTIO, WE ARE SECURE NOW!

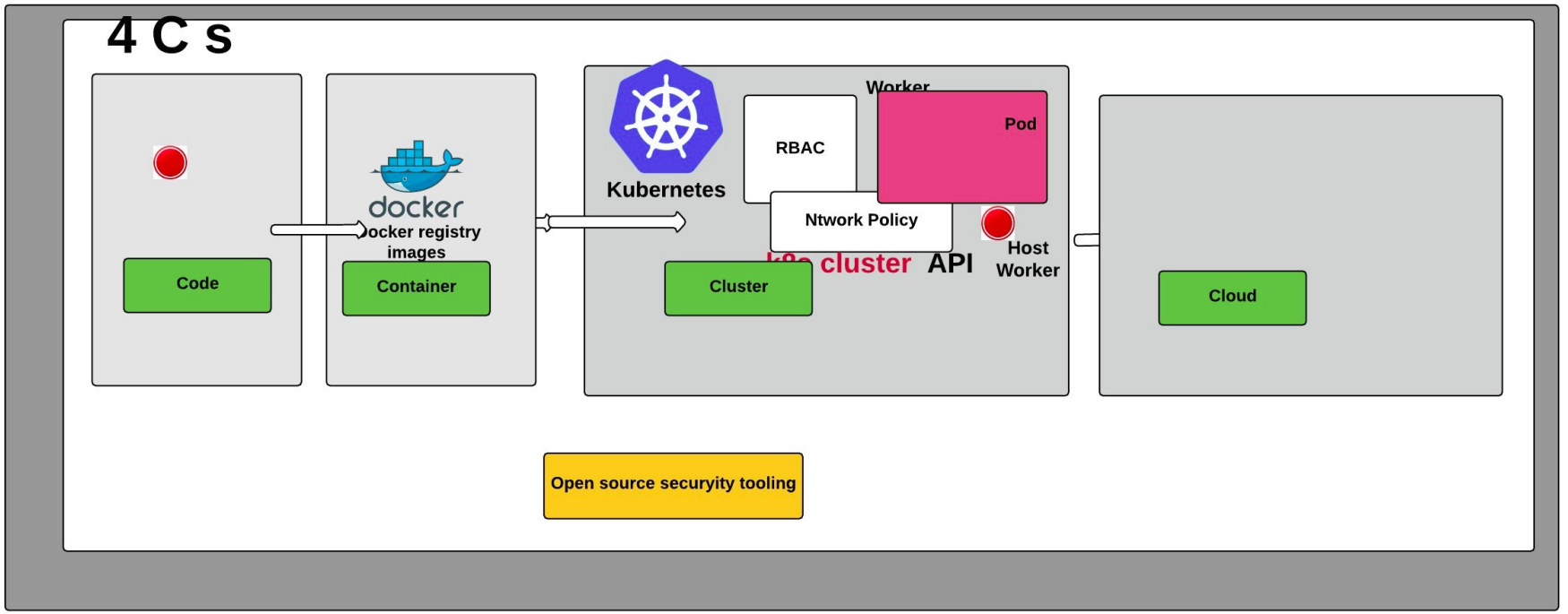


https://istio.io/docs/concepts/security/

```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  ingress:
  - from:
    -    namespaceSelector:
         matchLabels:
           project:
           myproject
    -    podSelector:
         matchLabels:
           role:
           frontend
    ports:
    - protocol: tcp
      port: 6379
```

an iptables like packet filter based on:

- Namespaces
- Labels
- Ports


4 C s
Code
docker
ocker registry
images
Container
Kubernetes
Worker
RBAC
Pod
Ntwork Policy
Host
Worker
cluster
API
Cluster
Cloud
Open source securyity tooling

Code

- Burpsuite
- Zap Proxy
- Static analysis
- Dynamic analysis
- Thread model

IMAGES

- Image Policy
- Registries
  - Clair, quay.io
  - Nexus
- ImageStreams

Kubernets

- Kube-hunter
- Kubiscan
- RBAC Audit tools
- KubeSec

More Kubernetes Issue

1. Storage
2. Images
3. Pod Security connected with linux kernel
4. Audit Logs
5. Networks
6. Databases
7. Helm security