RSAConference™2024
San Francisco | May 6 – 9 | Moscone Center

THE ART OF POSSIBLE

SESSION ID: SBX-R06

Kubernetes Security: Attacking And Defending Modern Infrastructure

#RSAC

Lenin Alevski
Security Engineer
Google
@alevsk

Max vonBlankenburg
Security Researcher
Semgrep
@hackfidgetcube

Disclaimer

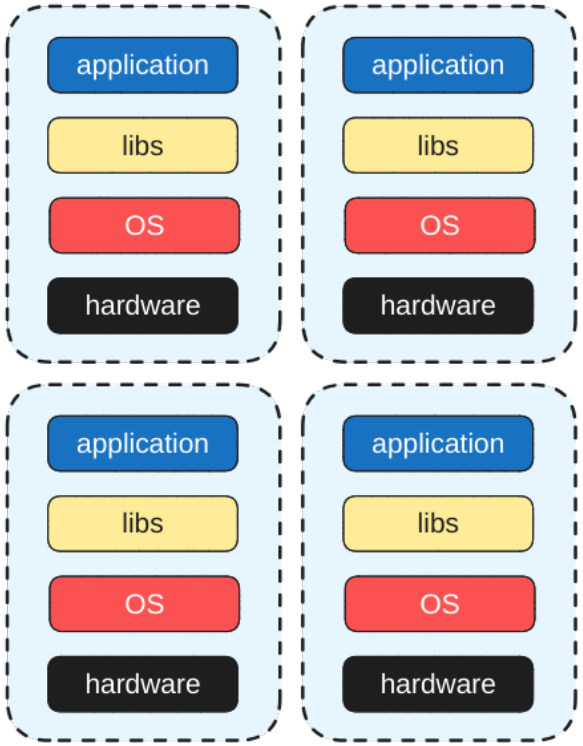
Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference™ or any other co-sponsors. RSA Conference does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

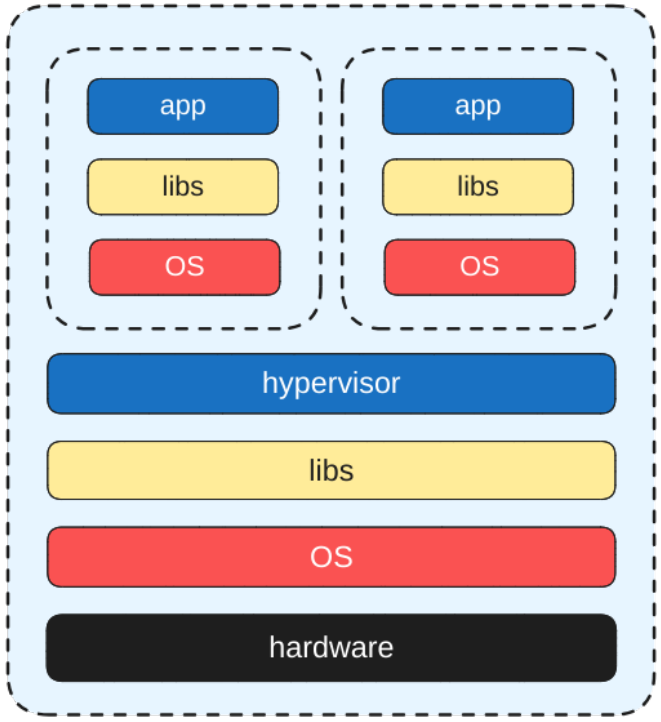
Agenda

- Introduction to Containers
- Introduction to Kubernetes
- Kubernetes Threat Model
- Most common attack techniques in K8S
- Build-In Defenses
- Kubernetes Evolution

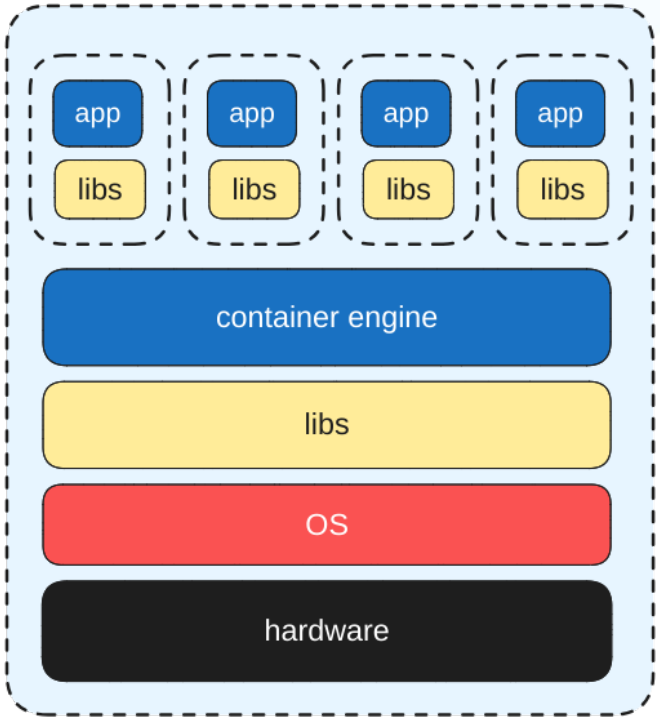
Application Compartmentalization



Traditional Architecture

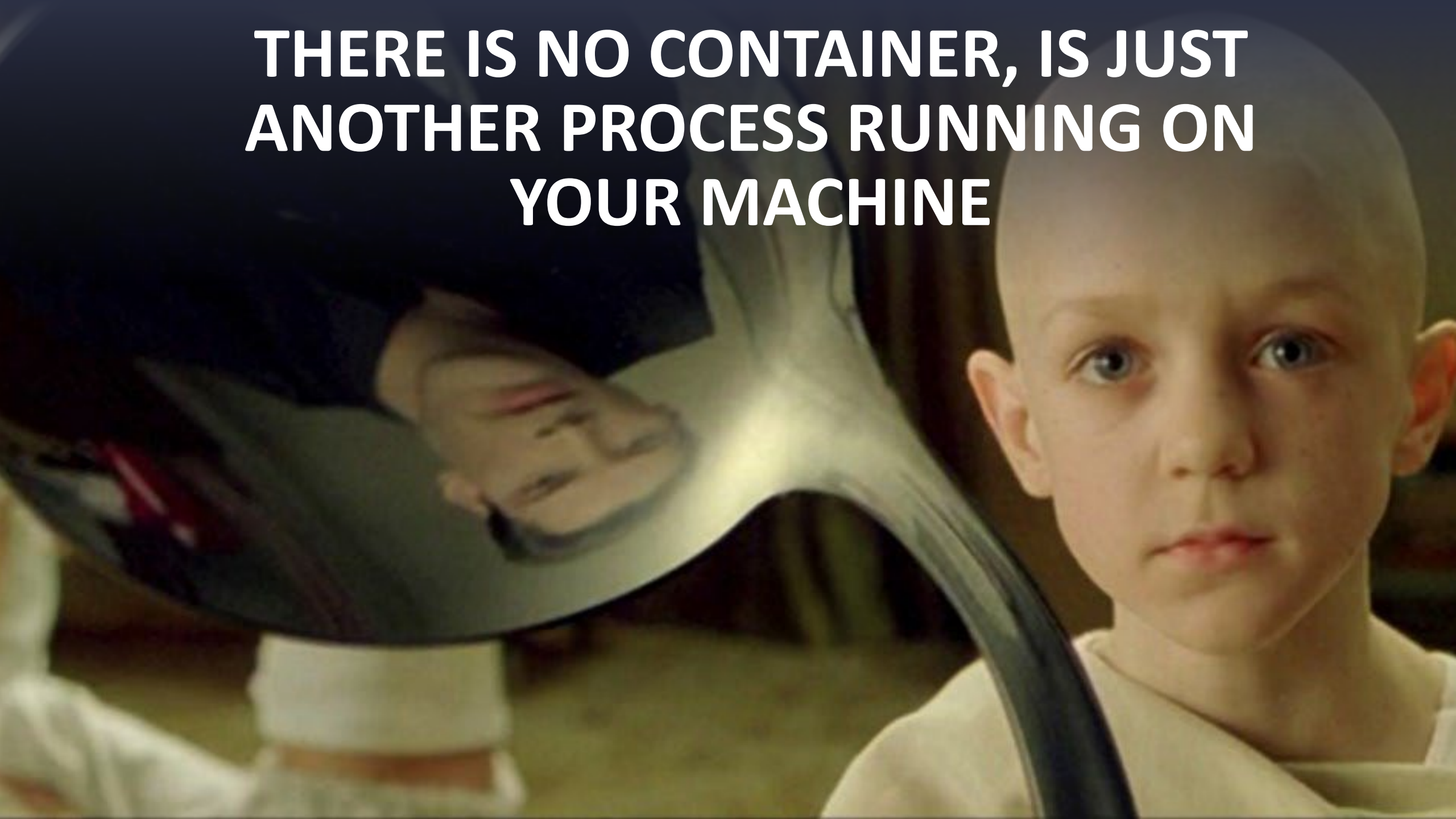


VM Architecture



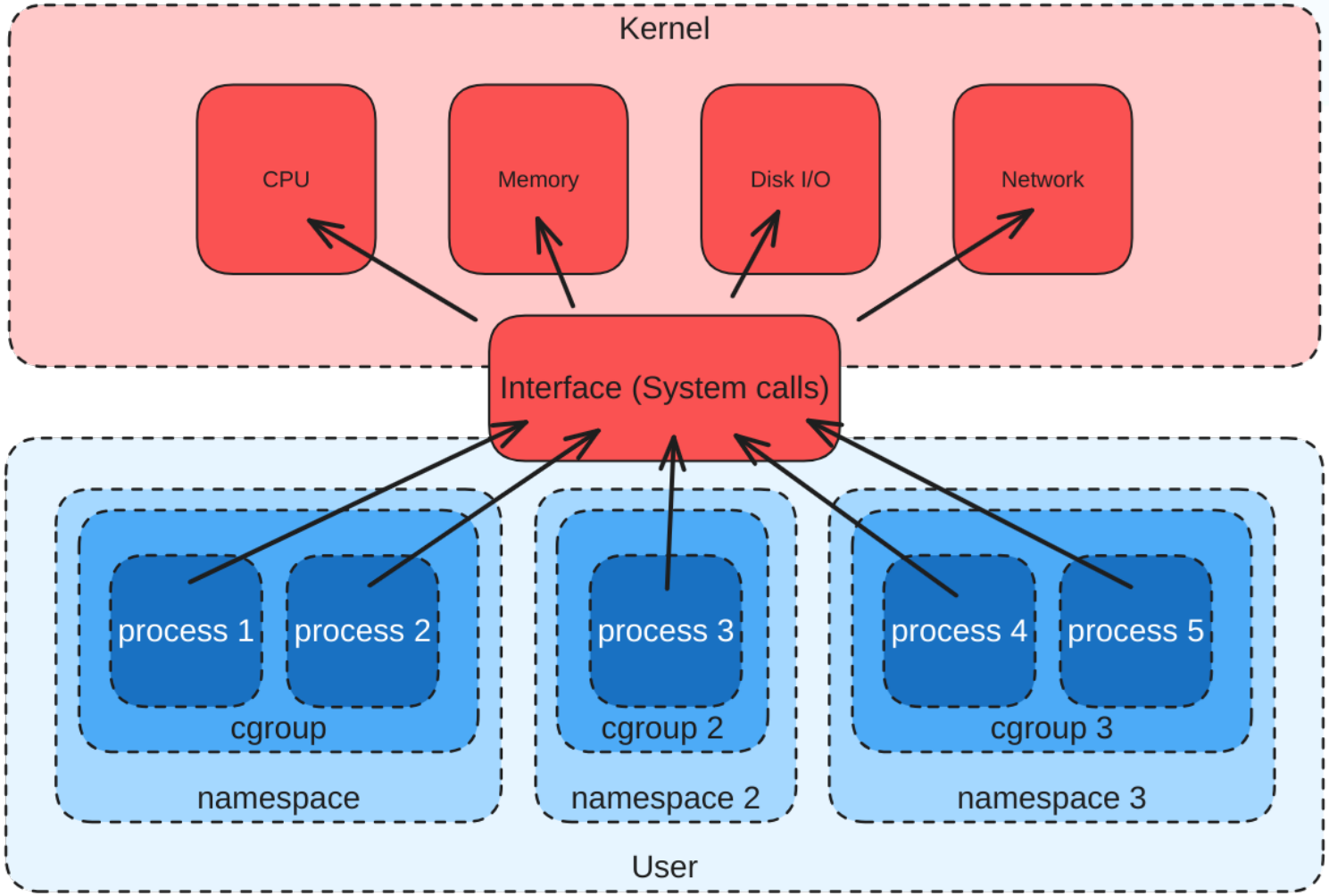
Container Architecture

THERE IS NO CONTAINER, IS JUST ANOTHER PROCESS RUNNING ON YOUR MACHINE

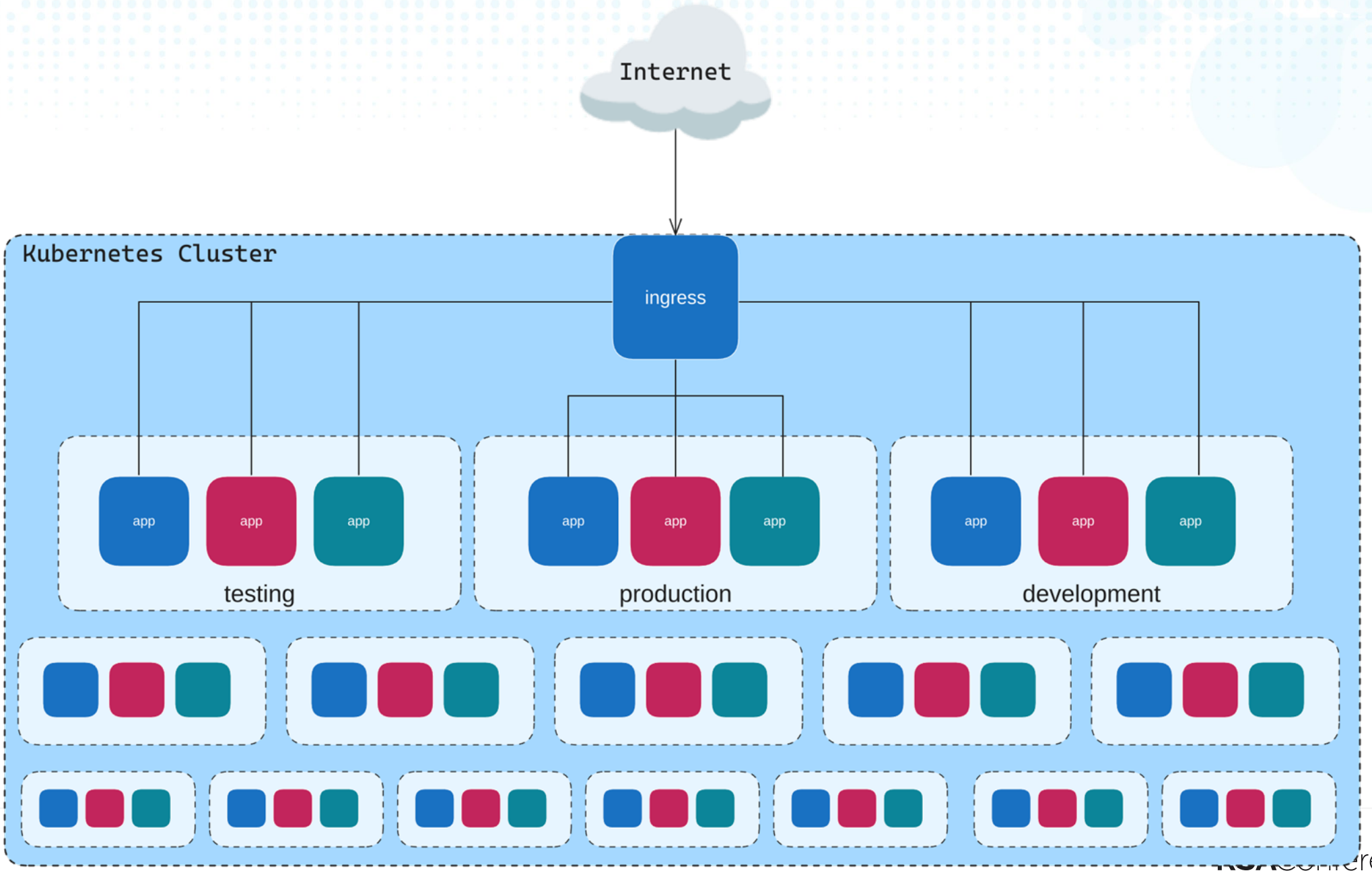


So, What Containers Really Are?

- Namespaces
- Cgroups
- Capabilities

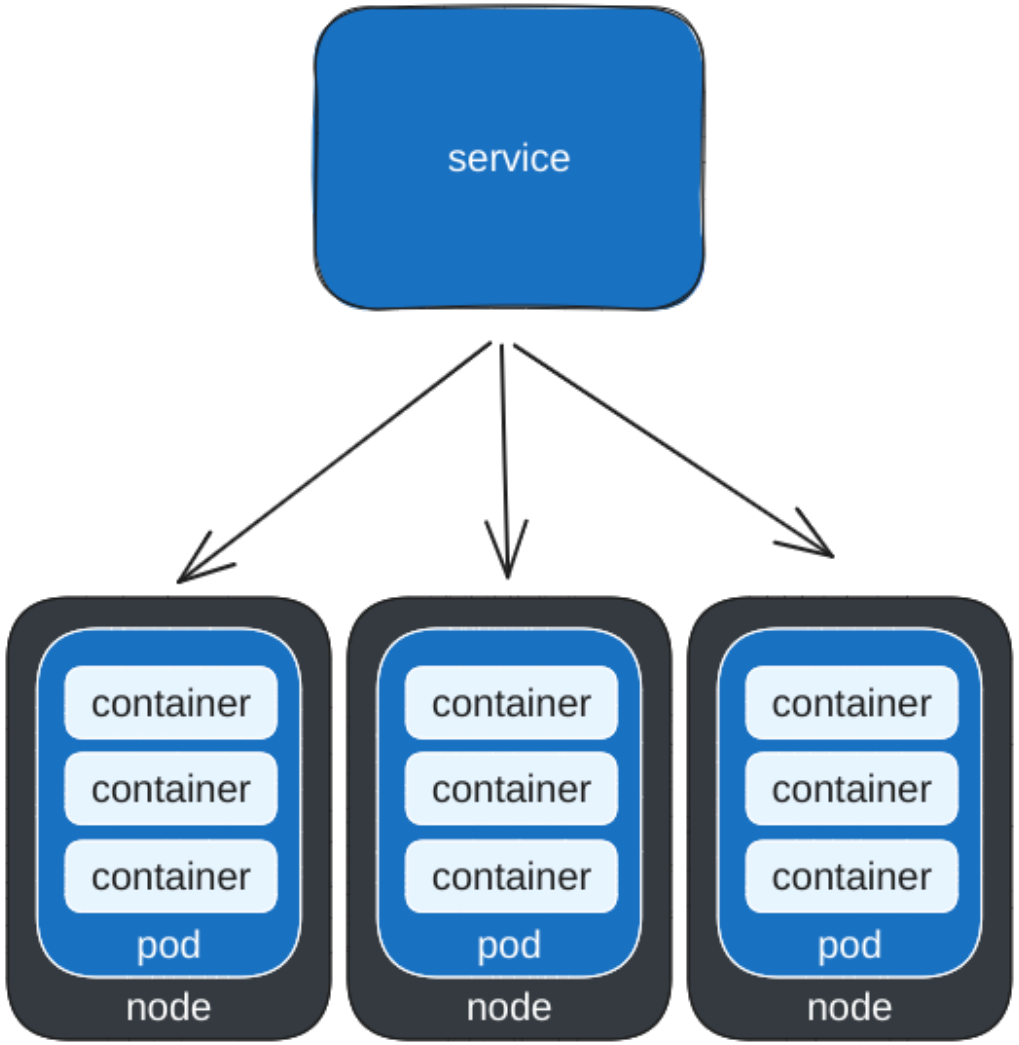


What Is Kubernetes Anyways?


Internet
Kubernetes Cluster
ingress
app
app
app
testing
app
app
app
production
app
app
app
development

Kubernetes Primitives

Kubernetes Application




```
apiVersion: v1
kind: Service
metadata:
  name: my-app
spec:
  selector:
   ...
---
apiVersion: apps/v1
kind: Deployment
metadata:
   ...
spec:
   ...
  Replicas: 3
  template:
   ...
  spec:
    containers:
      ...
....
....
```

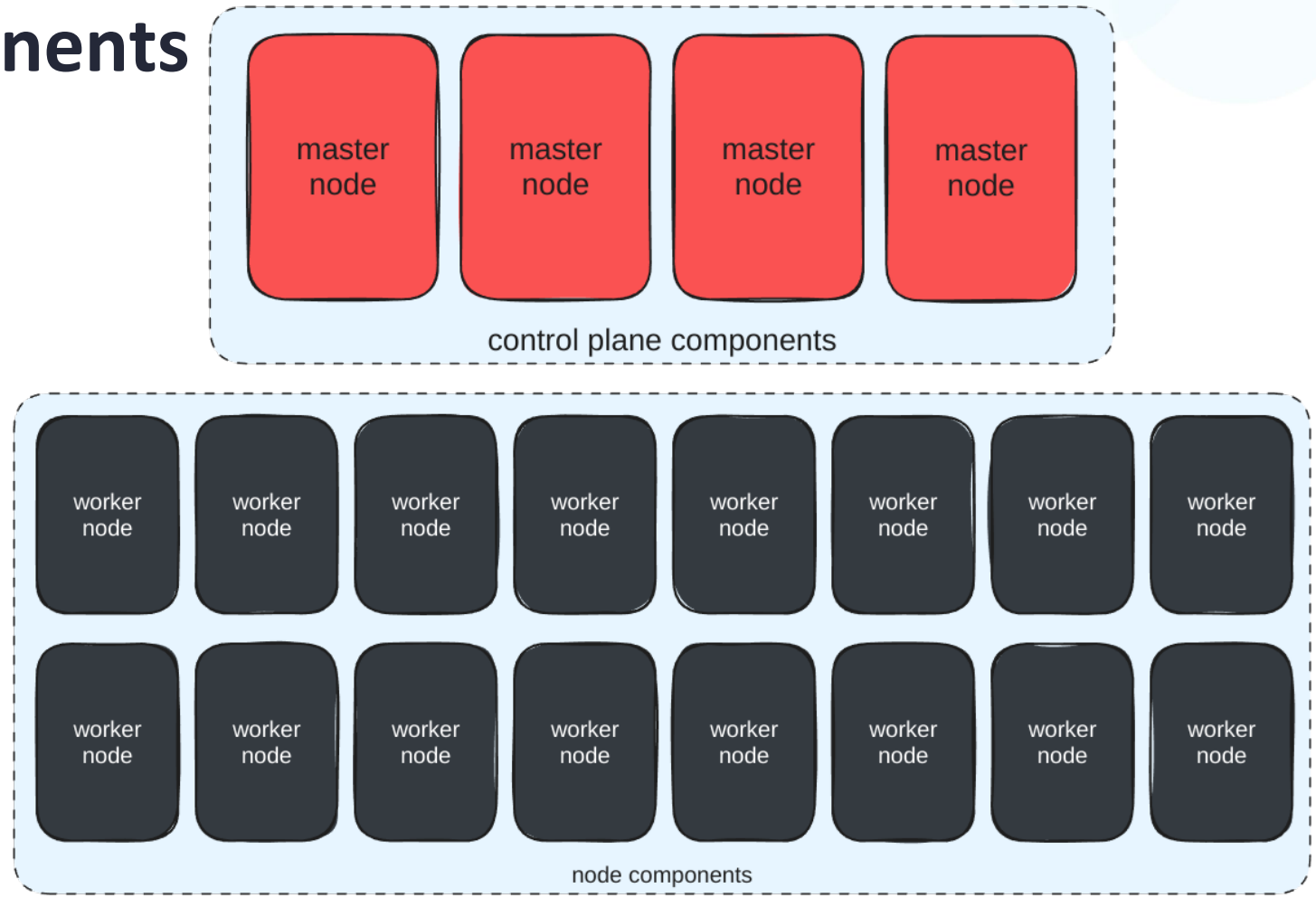
Kubernetes Components

Control Plane Components

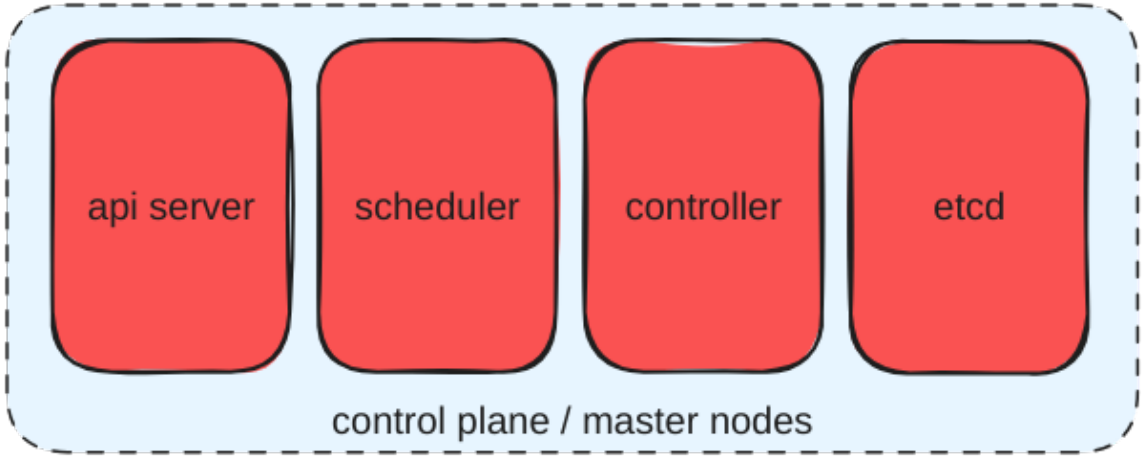
- kube-apiserver
- etcd
- kube-scheduler
- kube-controller

Node Components

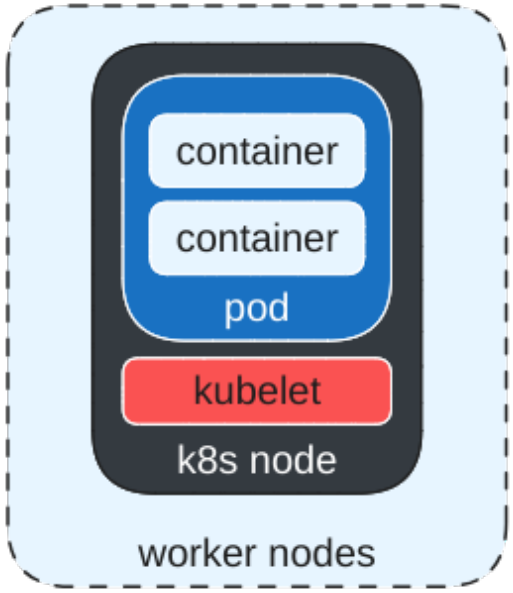
- kube-proxy
- kubelet
- container runtime



Kubernetes Threat Model: If An Attacker Controls…



Control plane nodes: Attacker controls your cluster. They can modify, access and destroy everything



Pod / Container: Attacker controls application and may be able to escape and attack the node

Kubelet: Attacker controls running pods

Worker nodes: Attacker controls running pods. They can attack master nodes

OWASP Kubernetes Risks Top (2022)

- K01: Insecure Workload Configurations
- K02: Supply Chain Vulnerabilities
- K03: Overly Permissive RBAC Configurations
- K04: Lack of Centralized Policy Enforcement
- K05: Inadequate Logging and Monitoring
- K06: Broken Authentication Mechanisms
- K07: Missing Network Segmentation Controls
- K08: Secrets Management Failures
- K09: Misconfigured Cluster Components
- K10: Outdated and Vulnerable Kubernetes Components

Kubernetes
Most Common Attack Techniques

Threat Matrix For Kubernetes (2020)

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	

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	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 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				Access managed identity credential	Instance Metadata API	Writable volume mounts on the host		
Exposed sensitive interfaces	Sidecar injection				Malicious admission controller		Access Kubernetes dashboard		
							Access tiller endpoint		
							CoreDNS poisoning		
							ARP poisoning and IP spoofing		

Threat Matrix For Kubernetes (2021)

= New technique

= Deprecated technique

Initial Access

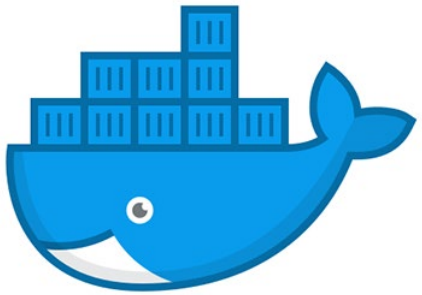
- Using cloud credentials
- Compromised images / registry
- Kubeconfig file
- Application Vulnerability
- Exposed sensitive interfaces

Using Cloud Credentials

Google Cloud

Compromised Registry And Images

- Supply Chain Attacks
- Vulnerable dependencies on images



Kubeconfig File - Hunting For .kube/config Files

raw.githubusercontent.com/cockpit-project/cockpit/944eded89ae9d5229b2dbd5463218e0a63beacfd/test/verify/files/openshift.kubeconfig

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM2akNDQWRLZ0F3SUJBZ0lCQVRBTkJna3Foa2lHOXcwQkFRc0ZBREFtTVNRd0lnWURWUVFEREJ0dmNHVnUKYzJocFpuUXRjMmxuYm1WeVF
UREQnR2Y0dWdWMyaHBablF0YzJsbmJtVnlRREUwT0RnNE1UUTBOVGN3CmdnRWlNQTBHQ1NxR1NJYjNEUUVCQVFVQUE0SUJEd0F3ZgFS0FvSUJBUURML2dEbFNmZnExK3NZMUNnbTZuZEYKOEVs
NPTW9IWFdXWnk5NVJPQ2UwREpGb3kyUTNncSs2WXVENi9mUE52c0RYSTFsZ0t1RTBneFZaaHJhCkFTNHJyRjRxdlJWRkhJQ3VtVy9nYUFYYnJla2N5d0NiRXBqM3hnL0pwamVUV2NvVHBtU0pSU
iN1ZnQgpVQm05dDQ1d1hnZ0JIM1VyUG5sTC9KK3psdm53NkdyaXRCUURjTmx0c2hxNEttUm0rUGt6RG1XTUxncXM1ZzRKCkFnTUJBQUdqSXpBaE1BNEdBMVVkRHdFQi93UUVBd0lDcERBUEJnTl
NUU4dm16VUYxb1B0YWV3UitMagpPQkVkSTNGZlFuUDZ1MmNFejl1em9VMlgvNzdMajEvTHk4RmpGMlJvZEdjVms5YXdQckl5SVIybnM4QUxNT0FBCnZkVXhSRWpXVWJUUjM3eGp1VmxicVRzeDF
GJjckx4cnl5Um5Xd0ZqRFZ3MURWS2I4WDZnWU9VOWRxYwp6ZkpBYUlHbXlkVTk1RkMwRTYza1ZhRy91RWZudFBhZjdBR3hjLzZXaHF3Z1RhZmk5OUxubUkrMmY4ZCtSZlM4CjAzelFONmh3ajhp
    server: https://10.111.112.101:8443
  name: 10-111-112-101:8443
contexts:
- context:
    cluster: 10-111-112-101:8443
    user: scruffy/10-111-112-101:8443
  name: /10-111-112-101:8443/scruffy
- context:
    cluster: 10-111-112-101:8443
    namespace: default
    user: system:admin/10-111-112-101:8443
  name: default/10-111-112-101:8443/system:admin
- context:
    cluster: 10-111-112-101:8443
    namespace: marmalade
    user: scruffy/10-111-112-101:8443
  name: marmalade/10-111-112-101:8443/scruffy
- context:
    cluster: 10-111-112-101:8443
    namespace: pizzazz
    user: scruffy/10-111-112-101:8443
  name: pizzazz/10-111-112-101:8443/scruffy
current-context: default/10-111-112-101:8443/system:admin
kind: Config
preferences: {}
users:
- name: scruffy/10-111-112-101:8443
  user:
    token: RmIp2HWbgoiMY8BbNxtodGUk3m1xjpK3jntHvq8yfrs
- name: system:admin/10-111-112-101:8443
  user:
    client-certificate-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUREVENDQWZXZ0F3SUJBZ0lCQ0RBTkJna3Foa2lHOXcwQkFRc0ZBREFtTVNRd0lnWURWUVFEREJ0dmNHVnUKYzJocFpuUXRjMmxuYm1WeVF
UtFeFZ6ZVhOMFpXMDZZMngxYzNSbGNpMWhaRzFwYm5NeEZUQVRCZ05WCkJBTVRESE41YzNSbGJUcGhaRzFwYmpDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUlKQU1z
c4ZG5YVGx0UXo0U2NKbXNNQlkvdTRjM0Y5UUFBNzJ1T3dhTGl1RjFWNzZDYzNyS1JLL2NidGs3CnNNMWFDdzJHK3pNc0ljT05tWUljNEFMeWNPMTdjTDZFRjIvOWZDZXN3bUlKeHE0QU4xdkt2T
PWXdQawpqRHh2dDJJdE9ab0FmYnFyUWQ5NGljMnYreTZMRTBVVDF2MGFqcDhadkE0bmdoaVNFNzhPcjhBMi9XWllDTi9BCkNvTFlHMnVQNmhmdEYwaU82OEFHU2JjQ0F3RUFBYU0xTURNd0RnWU
b1pJaHZjTkFRRUxCUUFEZ2dFQgpBSE1oMDI0WjFyRG1BeWFyV2g3OUVFZUFqenMxV1JkQlUrUWV6MXU3OWdsSnVFSjlPRW9mT0c5ZzVGOFpBdXFxCnJvSjhjdzBBZ2g2MlJDUTVCREJGZjRCRCt
VNmOHNjTUxFMWVWZ3dnK0NhMHp1b2Y5Tk5neXMyTXVEQQpLaGJWSTVyUmszTUdCN2lGTWQvK2tUU3lnUmdPWkhkWVA4a0lPWnhDOStZSWxzNGlqcGF0UnFWRjhaK1dZa2pOCjNzUEJPQ3pIV3Br
g0MzVSVHM9Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0K
    client-key-data:
LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBeXpXdWhLS3N2WXNqK3RsdWtuaW5XdGdiQkFJZEM3QjdsU25ScXFoRkorZUluZCtJCnRNMWRpMVl2aXZDWkZ
0xEWWI3TXl3aHc0MlpnaHpnQXZKdzdYdHd2b1FYYi8xOEo2ekNZZ25HcmdBMwpXOHE4NHk1dnREbGJaZUNoUGVMTjhheldQOVYyK0tNWk1DUk5TbVh4a2RHdUxyeEpsWTdRWEJ1RU83ZlV4VFk3
dEYjlabGdJMzhBS2d0Z2JhNC9xRiswWFNJN3J3QVpKdHdJREFRQUJBb0lCQVFDTXVrNXoyRnljbjJGNQpkVjdYM09xNVJQUGdvTDRRdkZXNWhVNklTeVFwY0hUeU1EM3lTRTdDRWh1MnNzeXhDW
XYnVaRUdBb0IKN1d3ajZXOVJXeWpOcmpEMEgzSkR3bDVhODQ5aWFIcmNLLzlRSTNFYTU1ci9YWndjc0kvaThFU1pFbnJoUEI4awo4NmwzV2EzVDFwM2lxYkFibzBzcGR0U3F4ZE8wQWtsNlh0Uj
UkI4ZTl6NnMxZlNiTXg2NnFIemQvWFMKYUloak5vZWhBb0dCQU9aYkRIUWJsM0VZMHNjUWh3cWpNYWlFN3A0VjNFMFVNY0VWdkpPNmZKNzVHMzNlZnlUego1MUgzcndrbG9Zcm5TSFRhcVUyZUE
zFSZ1R2a09Pb0IvMlhnbjl6cGpMUm9SQmV0bHQ1QW9HQkFPSFUKL3ErRS83OEFZUGovK01lVTZhRXNZVnNkSUNWczVQM1hnMjB6SDluVXhMMWJ0MnIwazc5SlRPangvSlV6VTBtdwpYSDhjVjRr
NTTVgyNUZjTGhyVVJaR01vZVorT2RkaldWS3ZBb0dCQUlTTERQNWR4ZVM0WU1kVGxqQWoKdUhYN2p3VDlGUlp3c1RRbFdxMHgxM1R0WnRrazFjbWdZWkJEVUw2Vm5YNjhGMi9OUlM4NDVYY3Fxa
pCm8xNnFsM1p0MEJPZnFwaUJFQWpJUm1zWkFvR0FJdnVEbUZ6WjRSU0Y4UXgxMVRUVUhBdHI2VmxScVhwODNuV0wKL2xaMUVLRk0rS2hFZVdUNW5pQVVzZUVLWjNKVDdJNU5SY04ySzdySFpubV
alExVUNLaHhESXN3aSs3CmRnb0Z4REVDZ1lCYklIa3NWK0NBbHFJdENFUjhHQVoreVlnNXNlVXFkQkIvNUtkRlFLMWgySDVGbldhaFkvd2cKdm1sS0pqd1IyekVTYWlsQlVHUHJURitNZktRRCt
1JBRlNyMjNnVUk0U0lURnVqVnc9PQotLS0tLUVORCBSU0EgUFJJVkFURSBLRVktLS0tLQo=
```

certificate-authority-data: LS0 /

Pull requests Issues Marketplace Explore

Repositories 0
Code 60
Commits 0
Issues 18
Discussions 0
Packages 0
Marketplace 0
Topics 0
Wikis 1
Users 0

Languages
Markdown 13
HTML 2
AsciiDoc 1
YAML 19
Text 9
HAProxy 2

Advanced search Cheat sheet

60 code results

Sort: Best match

vermegi/automate-private-AKS-AAD
/AKS-AAD-RBAC-NOTES.MD

```
13  ~~~yaml
14  apiVersion: v1
15  clusters:
16  - cluster:
17      certificate-authority-data: LS0tLS...LS0K (this is base64 encoded certificate data
    for the cluster)
...
59  - cluster:
60      certificate-authority-data: LS0...0tLS0K
61      server: https://xstofaks-79f2f498.hcp.westeurope.azmk8s.io:443
```

Markdown Showing the top three matches Last indexed on Jan 24

kspace5/svc-mesh-demo
cluster1/cluster-config.yaml

```
6       server: "https://ubsbonita/k8s/clusters/c-t95zn"
7       certificate-authority-data: "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJoekNDQ\
...
16        Uhtam54UjN1YWorcGhLWVI2VGtaM3V4QUlnTzI0WG5UeXdCQTBLQVRhS3k0b0NOTGpLNEsvdQp5S\
17        W1VcE9Mb2s3VkNGb2M9Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0=
18  - name: "kube1-ubsbonita"
19    cluster:
20      server: "https://192.168.86.68:6443"
```

YAML Showing the top two matches Last indexed on Apr 22, 2021

patiwat13/patiwat13
/kubeconfig-rancher.cfg

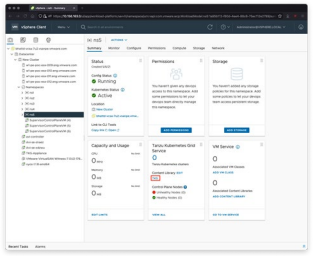
```
6       server: "https://203.151.50.20/k8s/clusters/c-bfhk6"
7       certificate-authority-data: "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJpRENDQ\
...
16        TQ2dWxkQVZHOC9pcWVkMDVYNU9iMFZYWFFDSURwWGNiUVV1c3NmZXROZm1IYjY1aHdjLzJ0VwpQO\
17        VYveHZ5T29rZUtBdlpkCi0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0=
18  - name: "immortal-cluster-k8s-master-2"
19    cluster:
```

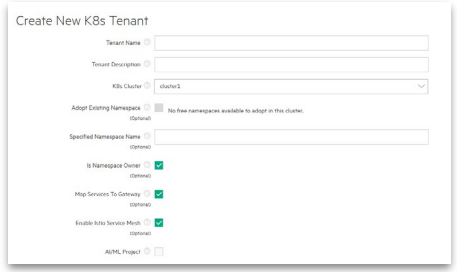
Application Vulnerabilities

- OWASP Top 10
- SQLi
- RCE
- Command injection
- Etc

A01:2021	Broken Access Control
A02:2021	Cryptographic Failures
A03:2021	Injection
A04:2021	Insecure Design
A05:2021	Security Misconfiguration
A06:2021	Vulnerable and Outdated Components
A07:2021	Identification and Authentication Failures
A08:2021	Software and Data Integrity Failures
A09:2021	Security Logging and Monitoring Failures
A10:2021	Server-Side Request Forgery

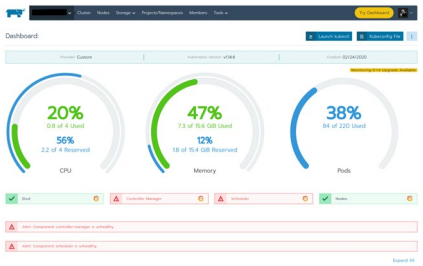
Exposed Sensitive Interfaces



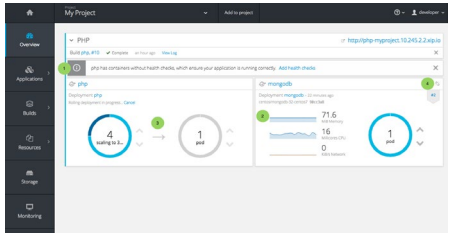


RANCHER

OPENSHIFT

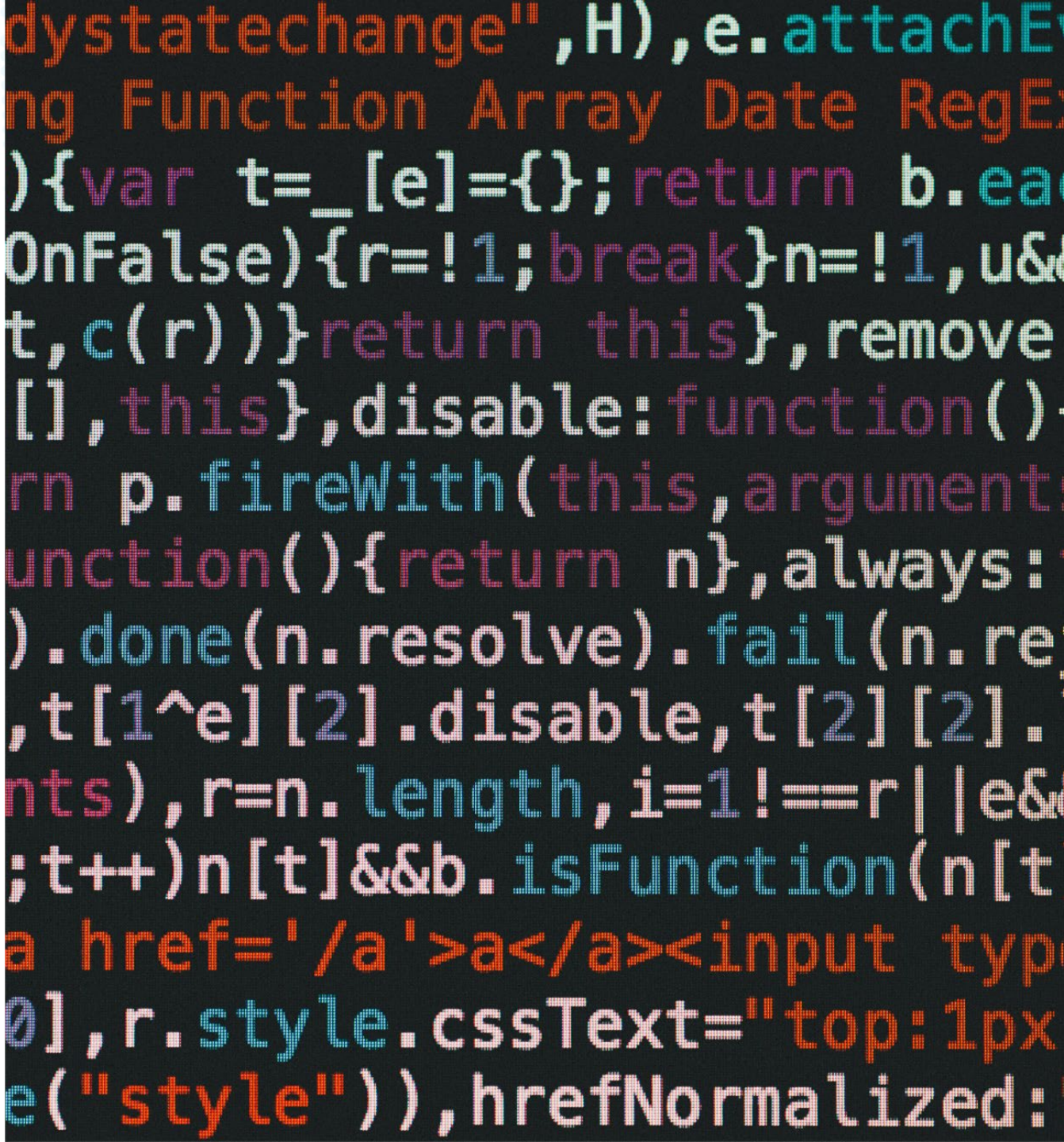


VMware Tanzu

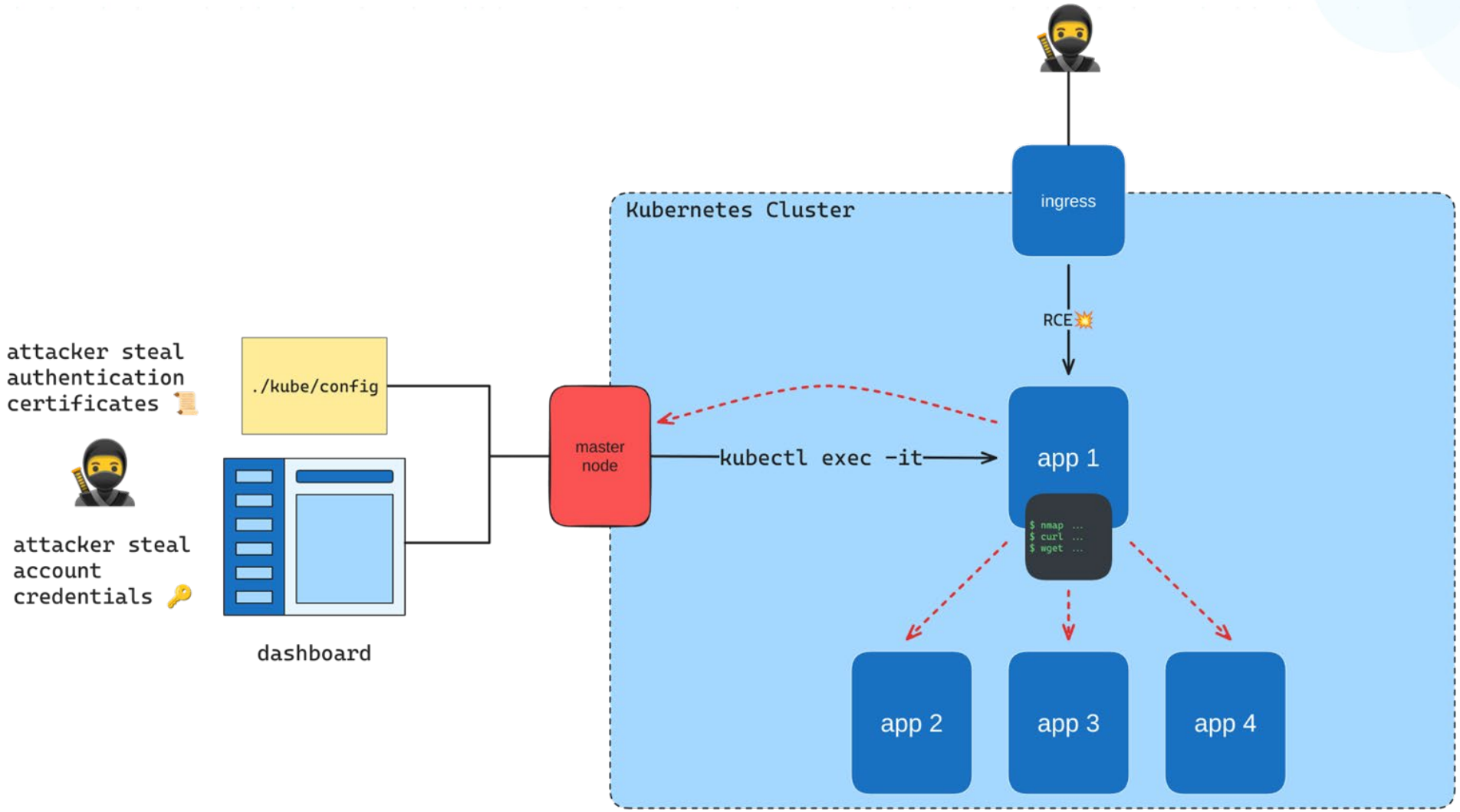


Execution

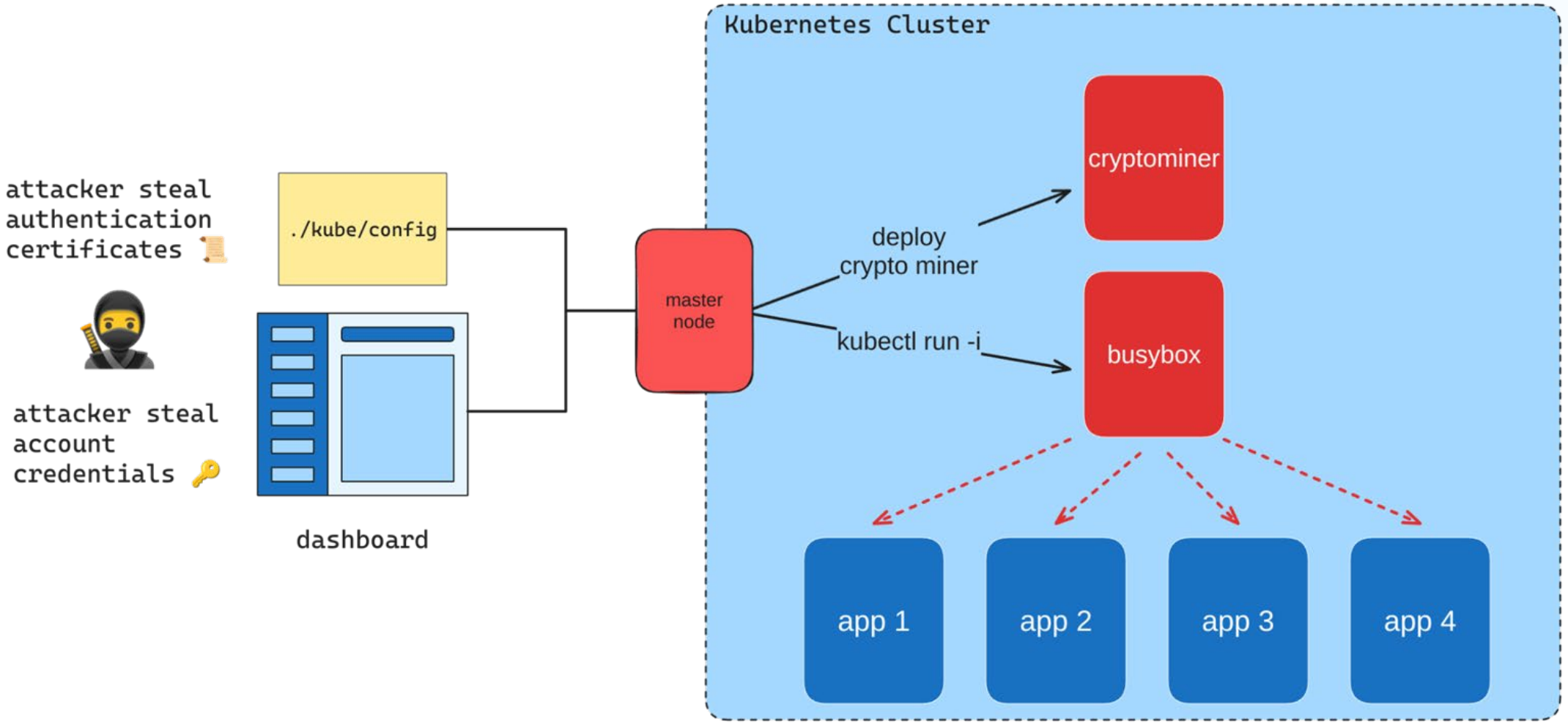
- Application exploit (RCE)
- Exec into container
- New container
- Sidecar Injection



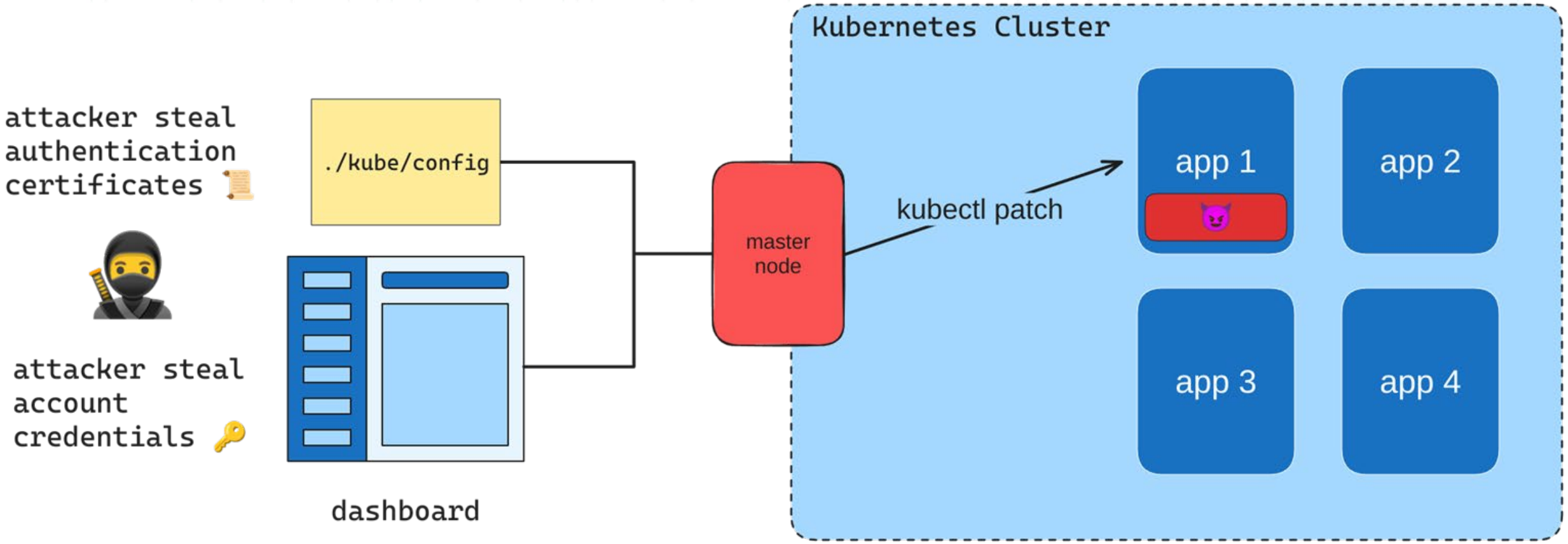
Exec Into The Container



Deploying New Containers



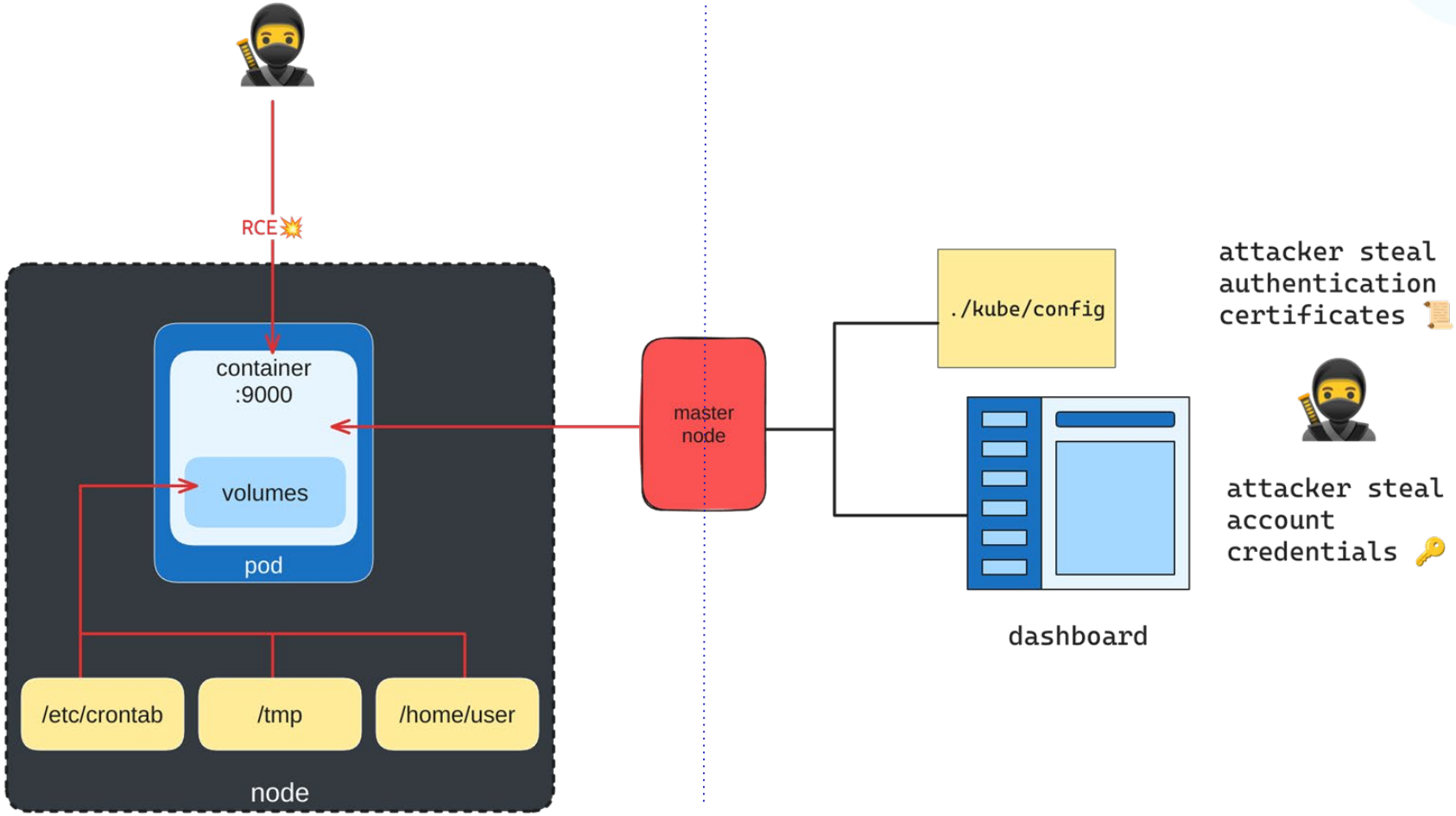
Sidecar Injection



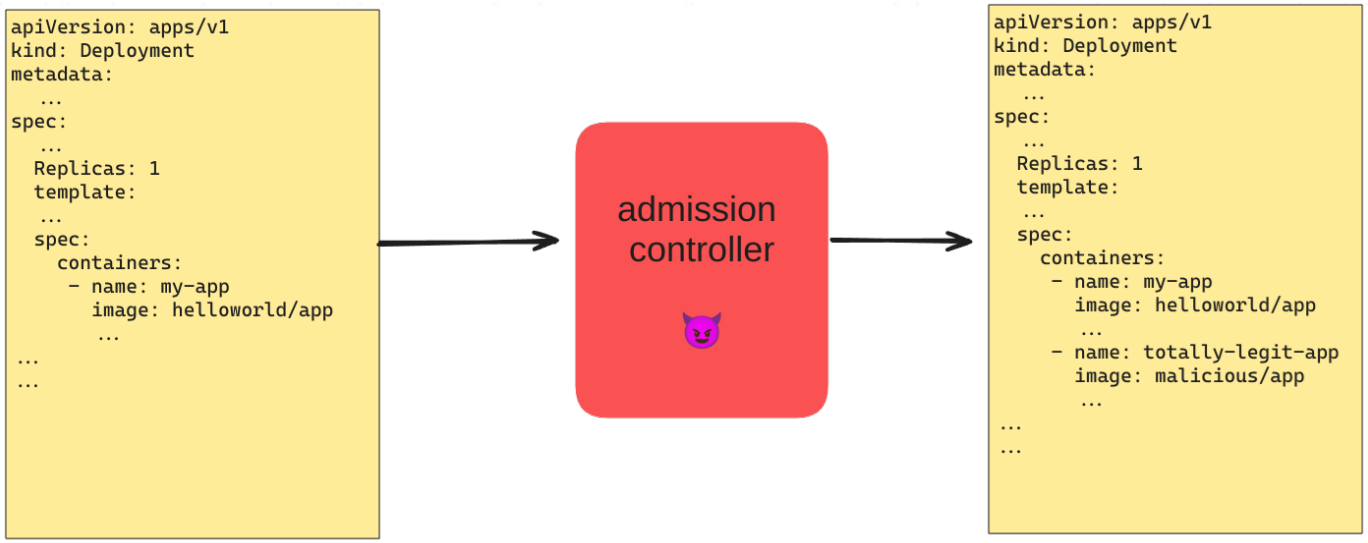
Persistence

- Backdoor container
- Kubernetes CronJob
- Writable hostPath mount
- Malicious admission controller

Writable Hostpath Mount



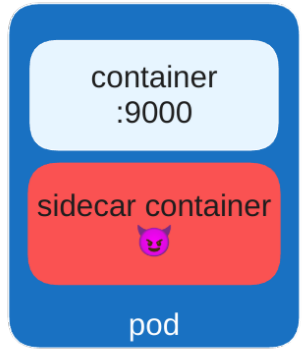
Malicious Admission Controller



Legitimate user deploying their workload

Attacker controlling admission controller

- malicious webhooks
- malicious mutation
- malicious validation
- tampered image



Privilege Escalation

- HostPath mount
- Access cloud resources
- Privileged container
- Cluster-admin binding

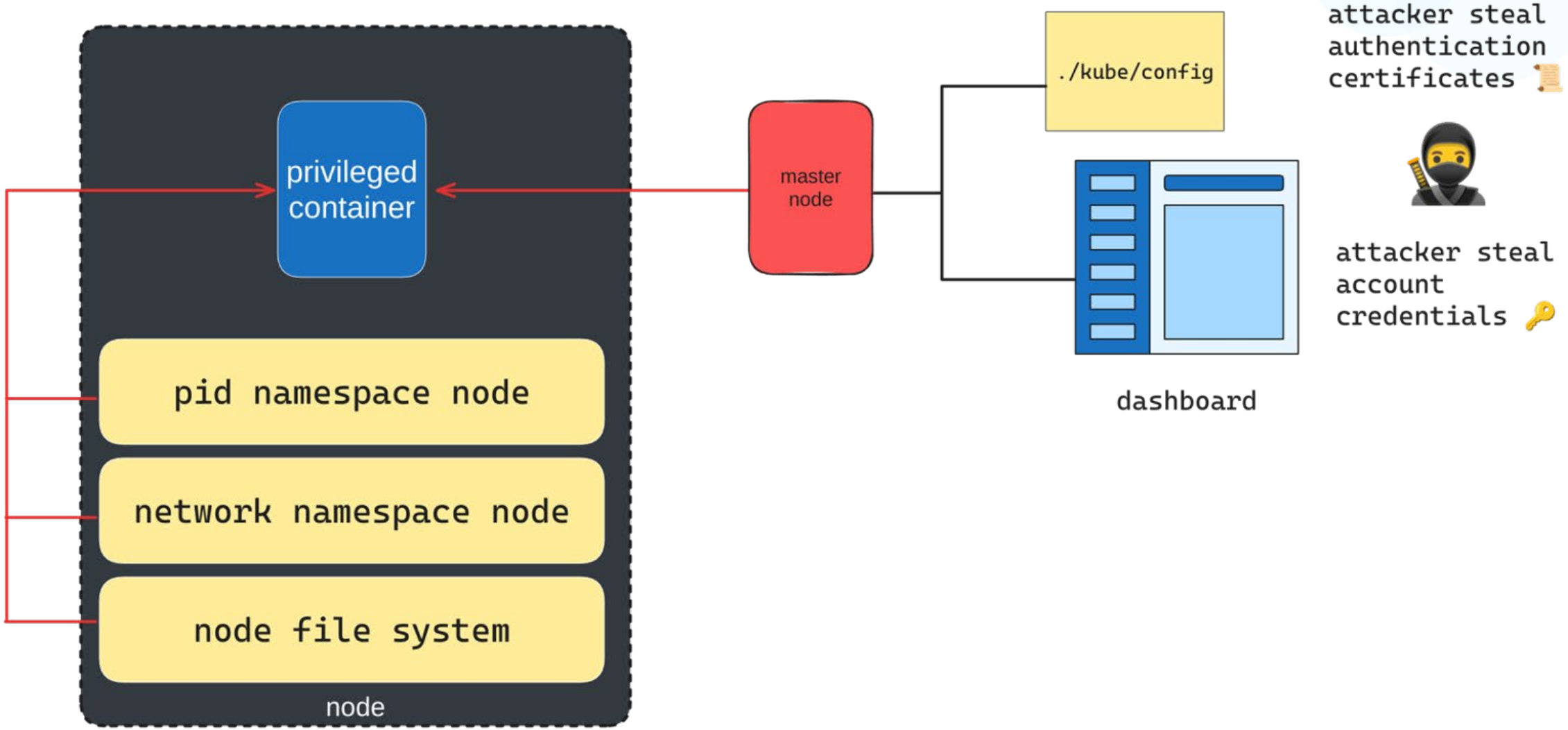
Privileged Container

Duffie Cooley 💉💉💉
@mauilion

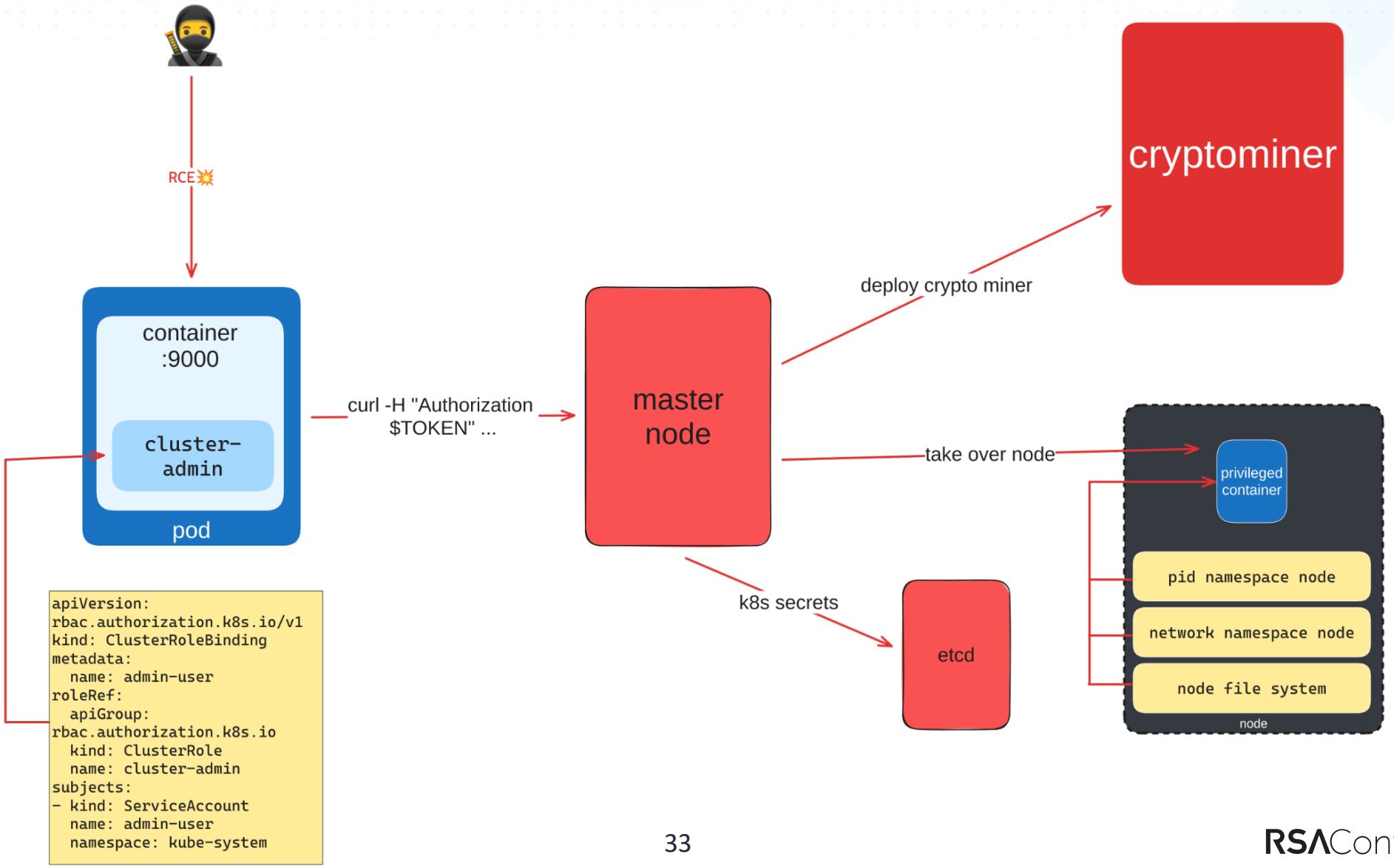
kubectl run r00t --restart=Never -ti --rm --image lol --overrides '{"spec":{"hostPID": true, "containers":[{"name":"1","image":"alpine","command":["nsenter","--mount=/proc/1/ns/mnt","--","/bin/bash"],"stdin":true,"tty":true,"securityContext":{"privileged":true}}]}}'

12:27 PM · May 17, 2019 · Twitter Web Client

A Container That Doesn't Contain Anything



Cluster-Admin Binding



Defense Evasion

- Clear container logs
- Delete Kubernetes events
- Pod / Container name similarity
- Connect from Proxy server

Credential Access

- List Kubernetes secrets
- Mount Service Principal
- Access container service account
- Applications credentials in configuration files
- Access managed identity credential
- Malicious admission controller

Discovery

- Access the Kubernetes API server
- Access Kubelet API
- Network mapping
- Access Kubernetes dashboard
- Instance Metadata API

Lateral Movement

- Access cloud resources
- Container service account
- Cluster internal networking
- Applications credentials in configuration files
- Writable volume mounts on the host
- CoreDNS poisoning

Collection

- Images from private registry

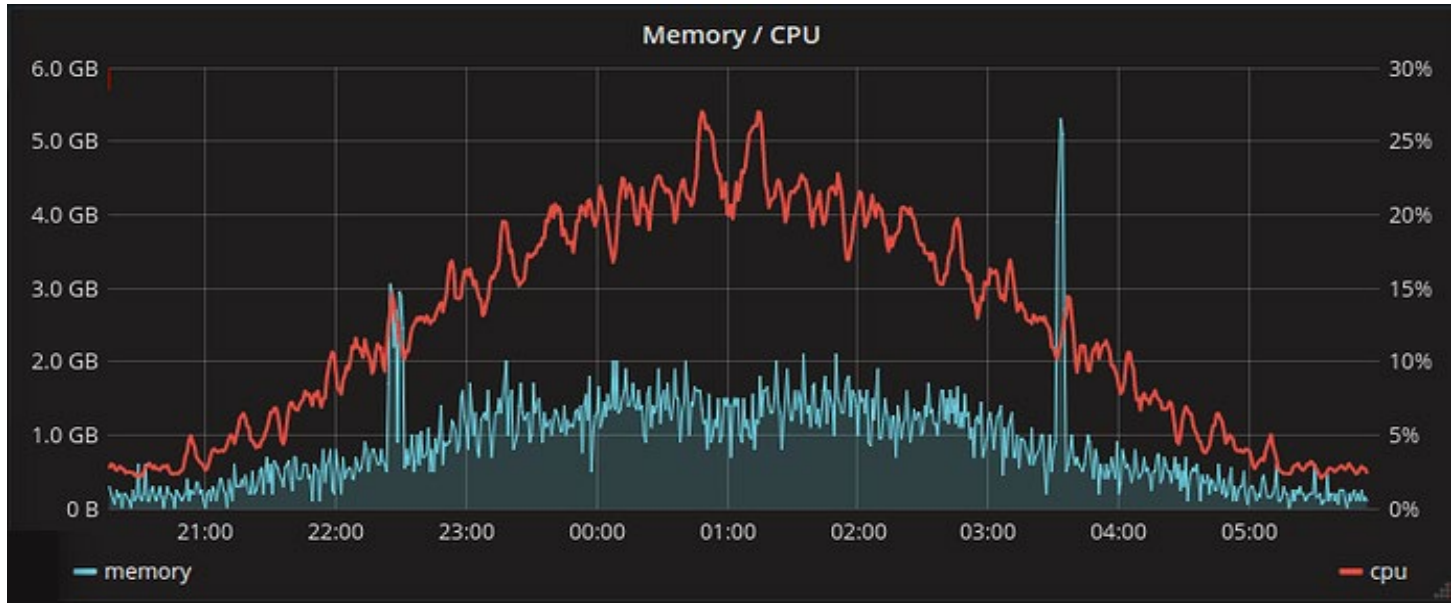
Impact

- Data destruction
  - Deleting PV, logs, events, etc
- Resource Hijacking
  - Deploying cryptominers, torrents, etc
- Denial of service
  - Deleting pods, deployments, statefulsets, services or ingress rules

Kubernetes Build-In Defenses

Resource Quotas and Limits

- Limits
  - CPU, Memory, etc.
- Requests
  - CPU, Memory, etc.



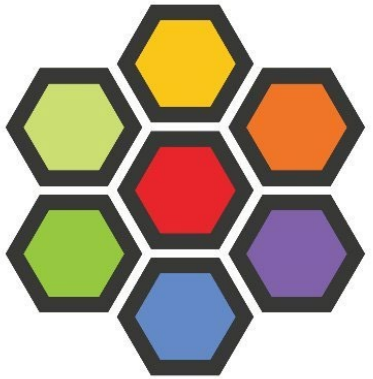
SecurityContext For Pods

- Privilege and Access Control
  - allowPrivilegeEscalation, privileged, runAsNonRoot, runAsUser, runAsGroup
- Capabilities Management
  - Capabilities: add, drop
- Filesystem and Mount Settings
  - readOnlyRootFilesystem
- Security Profiles
  - seLinuxOptions, seccompProfile, windowsOptions

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: sec-ctx-vol
    emptyDir: {}
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: sec-ctx-vol
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

Network Policies

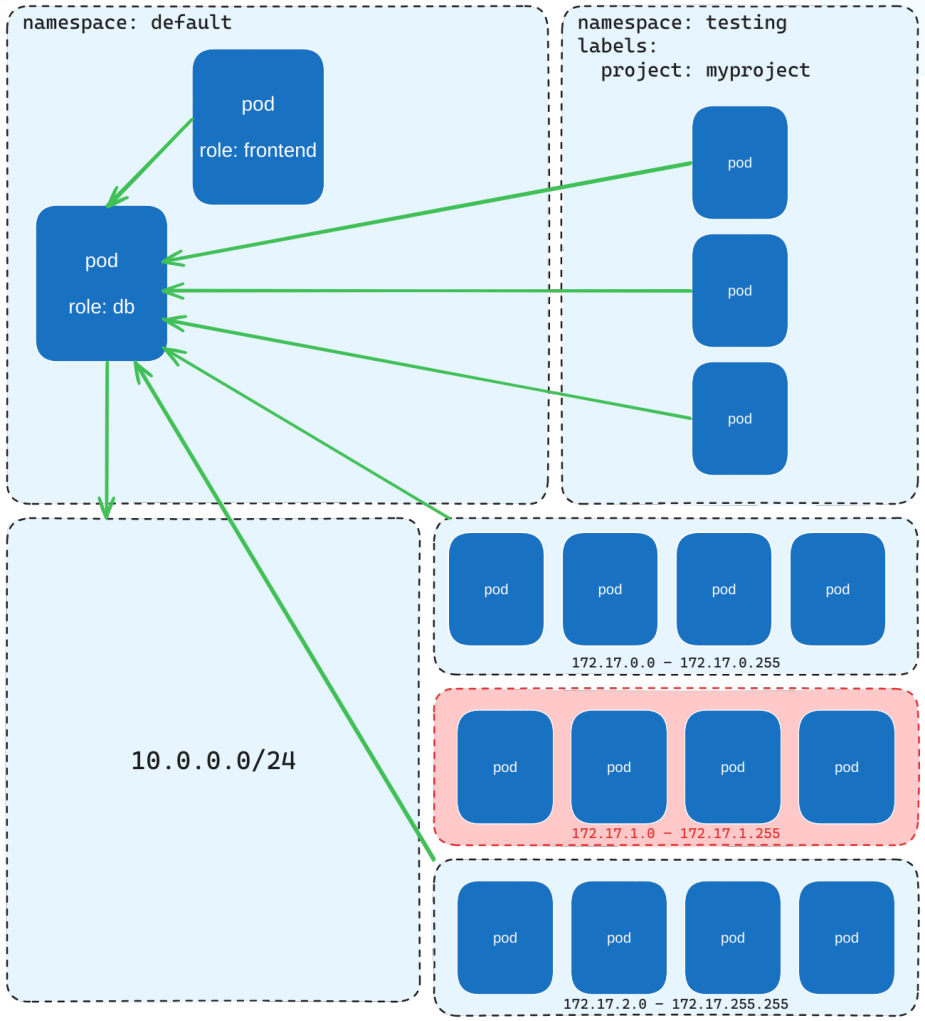
flannel



Istio

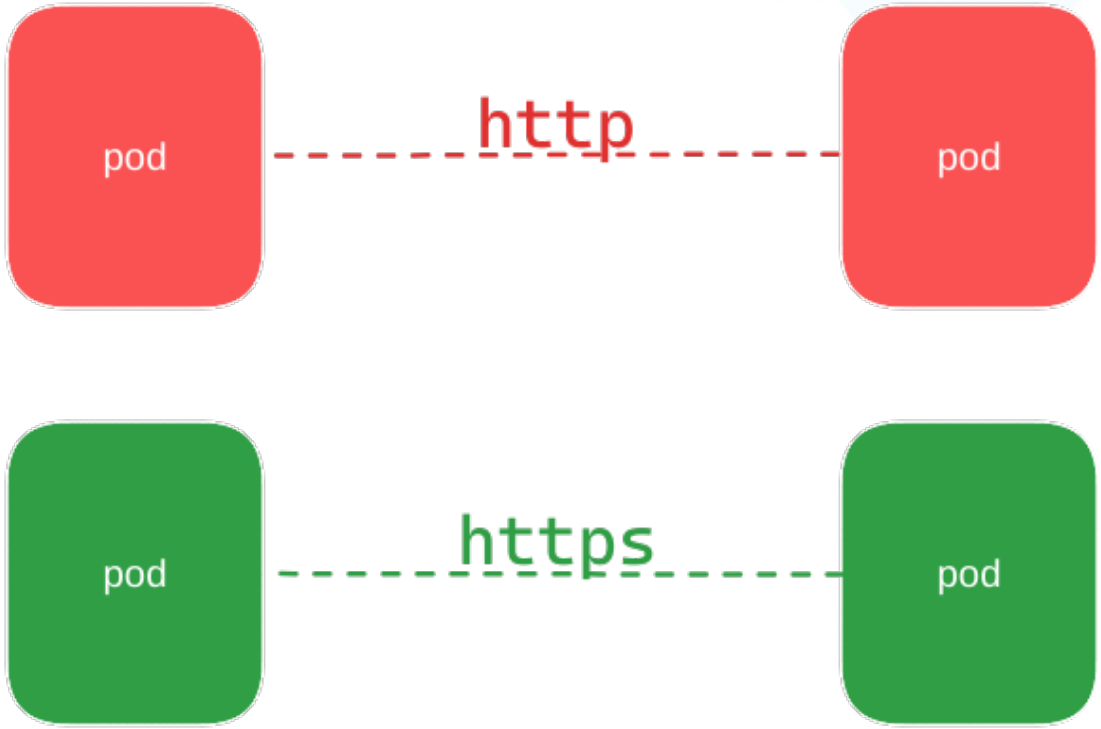
Network Policies

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
        - ipBlock:
            cidr: 172.17.0.0/16
            except:
              - 172.17.1.0/24
        - namespaceSelector:
            matchLabels:
              project: myproject
        - podSelector:
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
          port: 6379
  egress:
    - to:
        - ipBlock:
            cidr: 10.0.0.0/24
      ports:
        - protocol: TCP
          port: 5978
```



Encryption In Transit

- Kubernetes Certificate Authority
  - certificates.k8s.io
- Cert-manager.io
- Service mesh
  - Istio, Linkerd, Consul, etc



Encryption At Rest With EncryptionConfiguration

- Enable at-rest encryption for:
  - secrets, configmaps, events, *.apps, *.*, etc.
- Supports:
  - Identity (plain text), AESCBC, AESGCM
- External KMS

Pod Security Policies (PSPs)

- Deprecated in Kubernetes v1.21, and removed from Kubernetes in v1.25.

Protect Cluster Against Privilege Pods

PodSecurityPolicy

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: restrictive
spec:
  privileged: false
  hostNetwork: false
  allowPrivilegeEscalation: false
  defaultAllowPrivilegeEscalation: false
  hostPID: false
  hostIPC: false
  runAsUser:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  volumes:
  - 'configMap'
  - 'downwardAPI'
  - 'emptyDir'
  - 'persistentVolumeClaim'
  - 'secret'
  - 'projected'
```

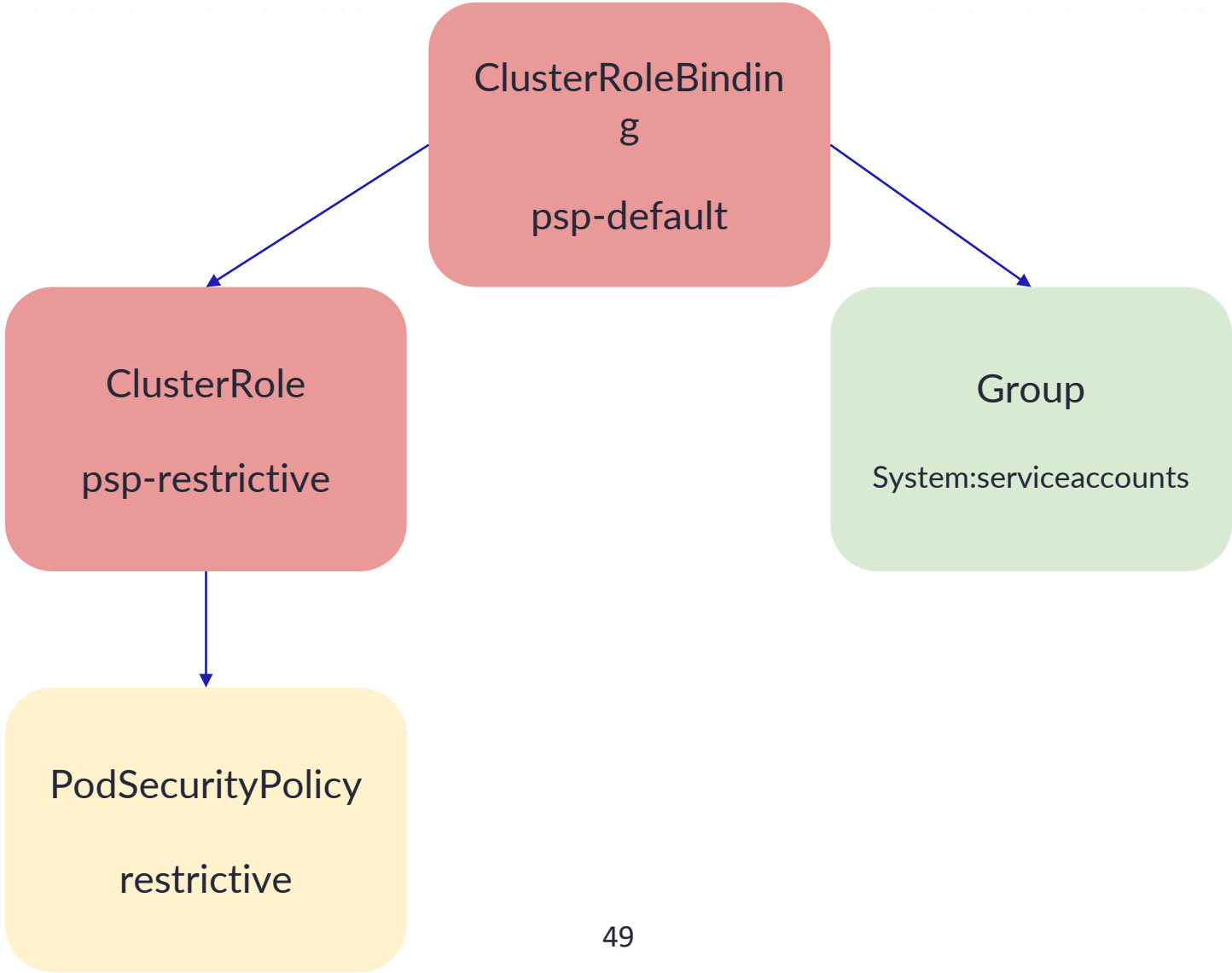
ClusterRole

```yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: psp-restrictive
rules:
- apiGroups:
  - extensions
  resources:
  - podsecuritypolicies
  resourceNames:
  - restrictive
  verbs:
  - use
```

ClusterRoleBinding

```yaml
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: psp-default
subjects:
- kind: Group
  name: system:serviceaccounts
  namespace: kube-system
roleRef:
  kind: ClusterRole
  name: psp-restrictive
  apiGroup: rbac.authorization.k8s.io
```

Protect Cluster Against Privilege Pods



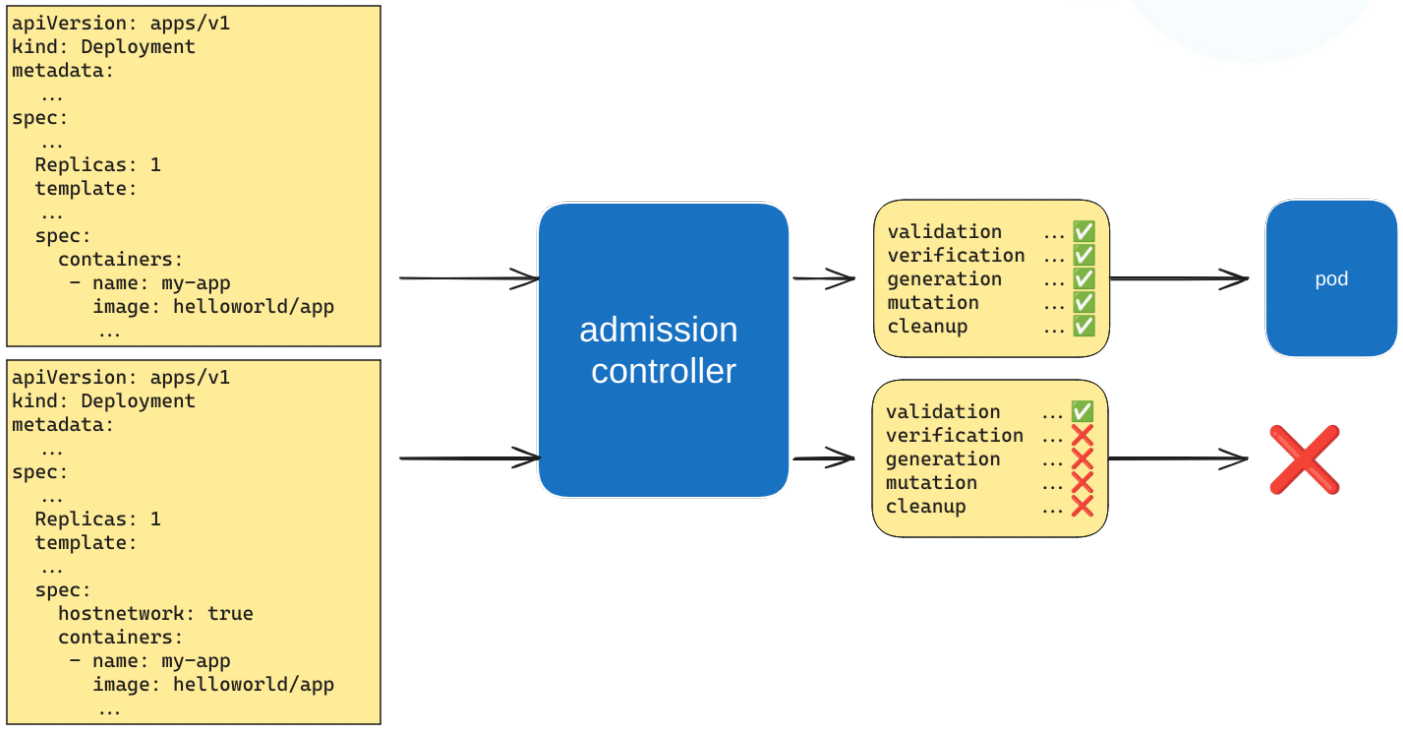
Protect Cluster Against Privilege Pods

```yaml
kind: Deployment
metadata:
  name: nginx-hostnetwork-deployment
  namespace: default
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.4
      hostNetwork: true
```

```
Controlled By:  Deployment/nginx-hostnetwork-deployment
Replicas:       0 current / 1 desired
Pods Status:    0 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=nginx
           pod-template-hash=597c4cff45
  Containers:
   nginx:
    Image:        nginx:1.15.4
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type             Status  Reason
  ----             ------  ------
  ReplicaFailure   True    FailedCreate
Events:
  Type     Reason        Age                From                   Message
  ----     ------        ----               ----                   -------
  Warning  FailedCreate  13s (x13 over 33s)  replicaset-controller  Error creating: pods "nginx-hostnetwork-deployment-597c4cff45-" is forbidden: unable to validate against any pod security policy: [spec.securityContext.hostNetwork: Invalid value: true: Host network is not allowed to be used]
```

Pod Security Admission

- Admission Controller
- Validation, mutation, cleanup Kubernetes resources
- OPA GateKeeper, Kyverno, jsPolicy, etc



Audit Log

- When to Log:
  - RequestReceived, ResponseStarted, ResponseComplete, Panic
- What to Log:
  - None, Metadata, Request, RequestResponse
- Where to Log:
  - Filesystem, Webhook

```
15 apiVersion: audit.k8s.io/v1 # This is required.
14 kind: Policy
13 # Don't generate audit events for all requests in RequestR
12 omitStages:
11   - "RequestReceived"
10 rules:
 9   # Log pod changes at Request level
 8   - level: Request
 7     resources:
 6     - group: ""
 5       resources: ["pods"]
 4   # Log pod changes at RequestResponse level
 3   - level: RequestResponse
 2     resources:
 1     - group: ""
 0       resources: ["pods"]
~
~
~
~
NORMAL audit-policy.yaml unix | utf-8 | yaml 100% 16:2
```

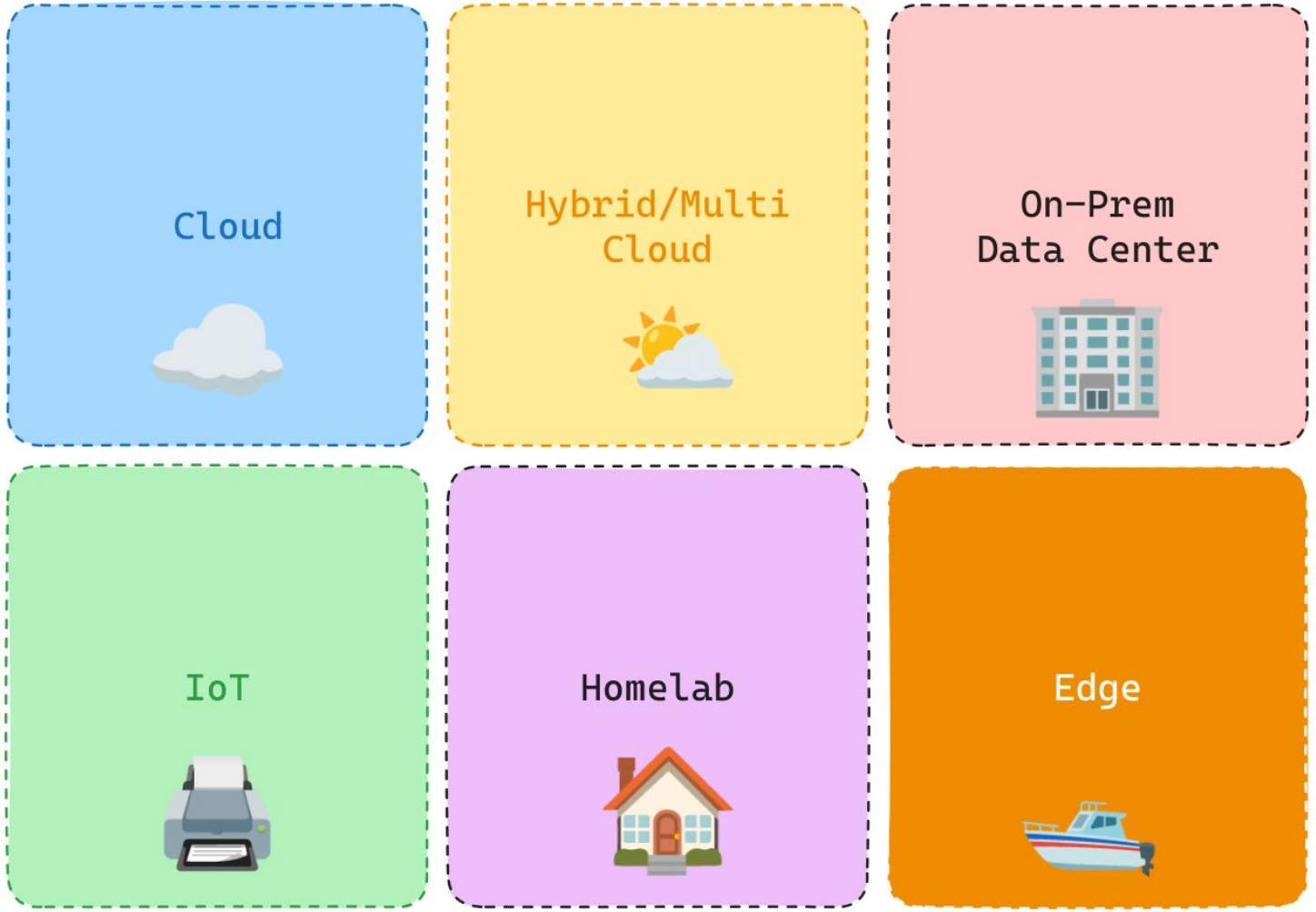
Kubernetes Cluster Hardening

- Isolate Kubernetes Nodes
  - Don’t expose nodes directly, use ingress controller, separate control and data flow
- Protect Kubelet
  - Disable anonymous authentication
  - Enable mTLS authentication on the kubelet's HTTPS endpoint
- Protect etcd with TLS, Firewall and Encryption
- Use Third-Party Authentication for API Server

What's Next?

Kubernetes Evolution (Edge & Hybrid Cloud)

- Kubeedge
- SuperEdge
- Akri
- OpenYurt
- K3s
- Microk8s
- Minikube
- …



Next Time You Deploy Kubernetes – Part 1

- Don’t deploy images from untrusted sources in your cluster, regularly scan images for vulnerabilities
- Don’t store sensitive data on kubernetes secrets, instead use an external KMS to manage secrets or enable EncryptionConfiguration
- Encryption in transit, generate TLS certificates for your workloads using the certificates.k8s.io api or use a third party solution like cert-manager.io

Next Time You Deploy Kubernetes – Part 2

- Enforce SecurityContext and limit resource consumption for your workloads
- Network Segmentation, choose CNI plugin that supports network policies to limit communication between your services and monitor network traffic
- Configure Role-Based Access Control (RBAC) policies via an Admission Controller or PSPs always following the principle of least privilege. Never use cluster-admin in your workloads

Next Time You Deploy Kubernetes – Part 3

- Turn on audit logging and monitor unusual or unwanted api calls, e.g. authentication failures.
- Keep Kubernetes version up to date, update and patch regularly
- Hardening control plane components and lock down kubelet
- Adopt GitOps practices for managing your Kubernetes clusters to enhance security, auditability, and consistency through declarative configuration, automation, and tight integration with version control systems

Thanks