



CKS Exam

Experience Topics & Tips



Who I am

Venkata Ramana Gali

- Senior Cloud DevOps Engineer/Architect



<https://www.linkedin.com/in/venkataramanagali>



<https://www.youtube.com/channel/UCwopwnnBoKMOUEOI6lefM0w/videos>



<https://github.com/ramanagali>

CKS Exam

- 100 Marks
- 2 Hours (**No Count down timer but Progress Indicator**)
- 16 Questions (between 15 - 20)
- v1.23
- Results in **24 hours**
- **No need of**
 - kubectl alias k
 - Bash Auto Completion
 - tmux, curl, wget, yq, jq
- Total of **16 Clusters**
- **Context Switch before every Question**

CKS In my view



- **Timing is very key ******
- More areas
- Security tools
- Hands on in syllabus and topics
- Modifying Control Plane Components
- Setup, Troubleshooting, Tracing etc
- Linux skills
- Speed Typing does matters (with accuracy)
- Careful with **YAML** Copy/Paste

Compare with CKA

- **CKA is all about**
 - API - Resources
 - Setup, Install & Backup
- **CKS is covers everything**
 - 4C's
 - Security setup deserves more
 - Tools & utilities
 - Deal with Control Plane
 - Linux skills
 - Sensitive YAML

Mock Exams

- Kodekloud 3 mock exams – Must do - each exam 4 to 5 times
- AcloudGuru 12 mock exams – Must do - 2 times
- Killer Shell Simulator sessions
 - – **Must do** – until confidence - before the exam
- <https://www.katacoda.com/>
- <https://killercoda.com/>
- <https://labs.play-with-k8s.com/>
- Local vagrant cluster (2 node)
 - <https://github.com/ramanagali/k8s-cluster>

CKS Exam 1st step



```
echo "set ts=2 sw=2 sts=2 expandtab" > ~/.vimrc
```

During the exam

- Do not type &&
- Copy paste the names, yaml etc
- Refer bookmarks

Use Documentation BookMarks



- [!\[\]\(97d7445697a94970d6443da16b12b5fa_img.jpg\) Kubectl Cheat Sheet](#)
- [!\[\]\(258b828d223cb6b35d2b001cf8c6c1e7_img.jpg\) Kubectl Commands](#)
- [!\[\]\(3ae3791b31be30fab239cd4ec63dfafe_img.jpg\) kube-apiserver KubeAPI Configuration](#)
- [!\[\]\(7835479178eea406b1967e30c6fcf845_img.jpg\) Kubelet Configuration \(v1beta1\)](#)
- [!\[\]\(bacea8e86e4e71bdd65916e2330bb152_img.jpg\) KubeConfig](#)
- [!\[\]\(103e1d41c49e54d33b8438f1f22779e7_img.jpg\) Install and Verify Binaries](#)
- [!\[\]\(127f84612f2d3af1475c40eb1d227d82_img.jpg\) Ingress with TLS](#)
- [!\[\]\(bd481c0ca3cf419dd12c334e7c6134ef_img.jpg\) NetworkPolicy NetPol](#)
- [!\[\]\(4bad89d7f2d54a9dd8d439405672b516_img.jpg\) Role RoleBinding Commands CLI RBAC](#)
- [!\[\]\(ca5efa1fd99f0257579e9142bacd2556_img.jpg\) POD volumemount \[hostPath\]](#)
- [!\[\]\(aa9ef994bed7fa38b15a4e5e70bf6ae9_img.jpg\) PSP PodSecurityPolicy](#)
- [!\[\]\(74566d81d00a0219b912a74e95a75e40_img.jpg\) Seccomp](#)
- [!\[\]\(943144c6978039e9c05c786b37ec3129_img.jpg\) AppArmor](#)
- [!\[\]\(9d3999dbe4ca37ffcfab8efdd4c45b35_img.jpg\) AppArmor Docs - GitLab](#)
- [!\[\]\(5ae02fd6274c2483590d6012e3d04afa_img.jpg\) Security Context](#)
- [!\[\]\(e5570f5f748ac88f4e736b3266532731_img.jpg\) ServiceAccount automountServiceAccountTo...](#)
- [!\[\]\(87b4db5879b0730071a0d46329036123_img.jpg\) Decode Secret](#)
- [!\[\]\(b08001ae99ef206fa4f9af57f20865f1_img.jpg\) Secret Types](#)
- [!\[\]\(854c302f4c2b1fa304f37352e1f6040c_img.jpg\) Secret as VolumeMount](#)
- [!\[\]\(db2e055846185a85e7727455d0ebb2c6_img.jpg\) Image Policy Webhook - IPW](#)
- [!\[\]\(f94e2b1e8b460412680111ec8ab42956_img.jpg\) Audit Policy](#)
- [!\[\]\(e1994e58386436a56e274b4794576284_img.jpg\) RuntimeClass](#)
- [!\[\]\(c297745f2265c49f12f422c1d0e3faf3_img.jpg\) Runtime Class Examples - GitHub](#)
- [!\[\]\(88ebdcdd4a6cf8af176ba422814aa6a7_img.jpg\) Pod Level Logging with Kubernetes](#)
- [!\[\]\(78d629f5c51fa974fa2b686e68b5e10a_img.jpg\) Trivy Examples](#)
- [!\[\]\(79d1eb3b4ad4722eca75d35d916a40b9_img.jpg\) Trivy - GitHub](#)
- [!\[\]\(ea49966dec3db5f47646d404956733d2_img.jpg\) Falco Supported Conditions Outputs](#)
- [!\[\]\(aac249bef4c11a70843828fb0e5ba317_img.jpg\) FALCO: default rules](#)
- [!\[\]\(ecf7706a2c8cac479935df99b83367a2_img.jpg\) Examples](#)
- [!\[\]\(c7bd4a450cb0cd714108d844e86a4c8a_img.jpg\) K8S Binaries and Release Notes - GitHub](#)
- [!\[\]\(4d41dd10235d2e3427042e475a7dc346_img.jpg\) crioctl](#)
- [!\[\]\(a7fc99238206aecc8f2525aa42d56d25_img.jpg\) Access API Server on Two Ports](#)

CKS Exam Categories



- 1. Control Plane related (API Server, etcd, kubelet etc)**
- 2. General Kubernetes (Resources)**

Control Plane related



1. CIS Benchmarking
2. PodSecurityPolicy
3. ImagePolicyWebhook/AdmissionController
4. AuditPolicy
5. Falco/Sysdig
6. Tracing Container syscall
7. Hardening access to Kubernetes API
8. Troubleshoot API Server

General Resource Related

1. NetworkPolicy
2. RBAC
3. Secret
4. ServiceAccount
5. AppArmor
6. SecComp
7. RuntimeClass
8. SecurityContext & Immutable Pods
9. Static Analysis
10. Trivy
11. Ingress TLS
12. OPA

1. CIS Benchmarking

- Given CIS Benchmark Scan results
- Fix based on its remediations (displayed along with result)
- Control Plane Component(s) issues
- Kubelet Config issues etc

Refer: <https://downloads.cisecurity.org/#/>



2. PodSecurityPolicy

```

apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: example
spec:
  privileged: false # Don't allow privileged pods!
  # The rest fills in some required fields.
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  volumes:
    - '*'

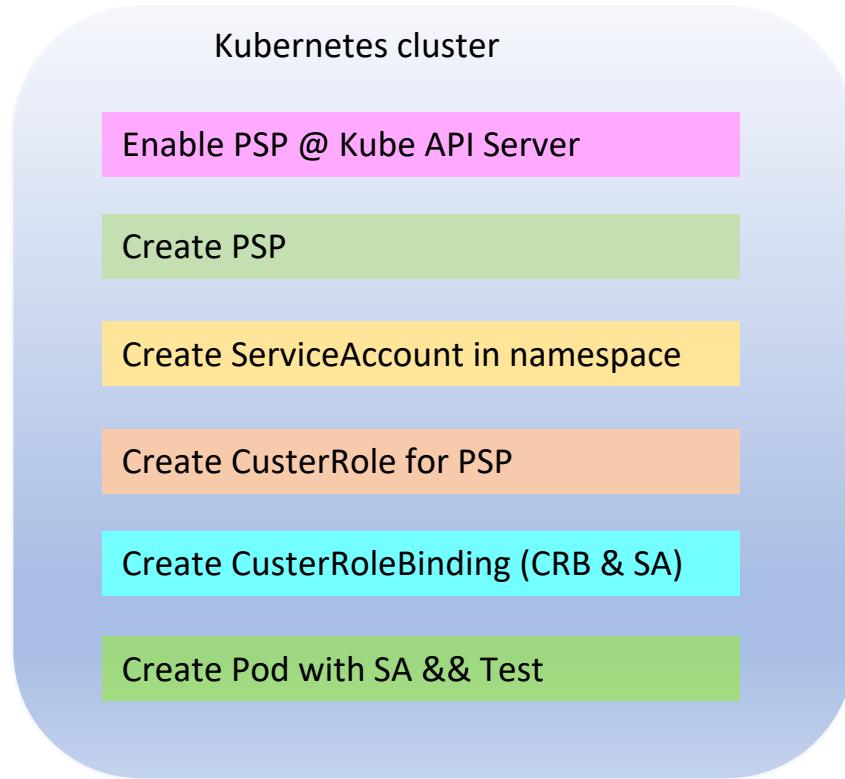
```

```

apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: privileged
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
    - '*'
  volumes:
    - '*'
  hostNetwork: true
  hostPorts:
    - min: 0
      max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'

```

2. PodSecurityPolicy Implementation



Kube-API-Server level...

--enable-admission-plugins=NodeRestriction, **PodSecurityPolicy**

3. ImagePolicyWebhook

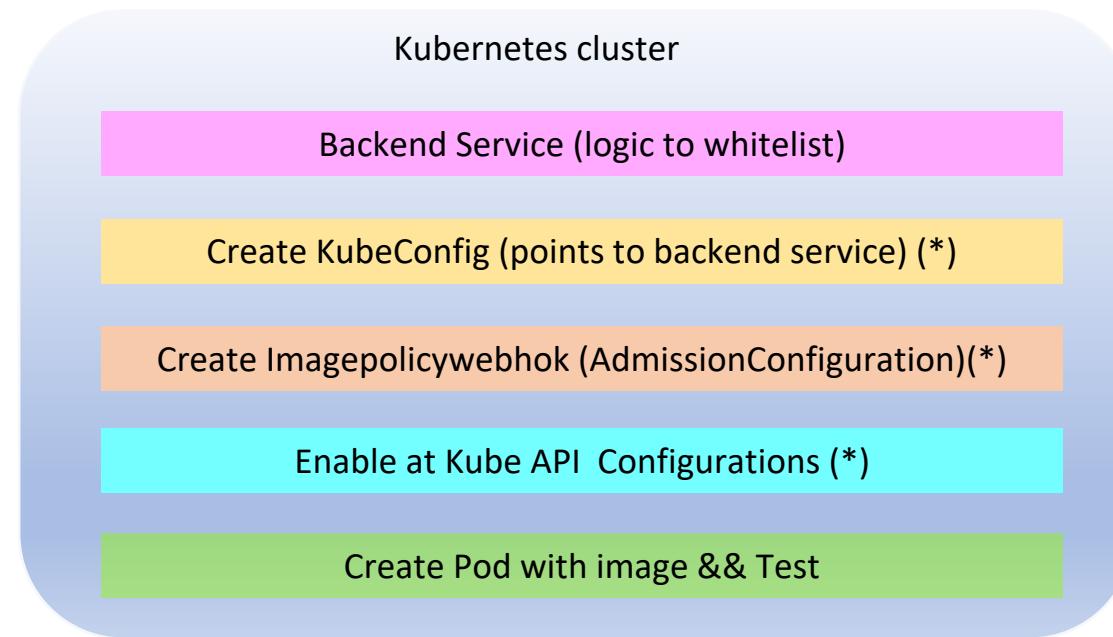
/etc/kubernetes/xyz/admission_kube_config.json

```
{  
  "imagePolicy": {  
    "kubeConfigFile": "/etc/kubernetes/pki/webhook/admission_kube_config.yaml",  
    "allowTTL": 50,  
    "denyTTL": 50,  
    "retryBackoff": 500,  
    "defaultAllow": false  
  }  
}
```

```
apiVersion: apiserver.config.k8s.io/v1  
kind: AdmissionConfiguration  
plugins:  
- name: ImagePolicyWebhook  
  configuration:  
    imagePolicy:  
      kubeConfigFile: <path-to-kubeconfig-file>  
      allowTTL: 50  
      denyTTL: 50  
      retryBackoff: 500  
      defaultAllow: true
```

```
apiVersion: v1  
kind: Config  
clusters:  
- cluster:  
  certificate-authority: /etc/kubernetes/pki/server.crt  
  server: https://image-bouncer-webhook:30080/image_policy  
  name: bouncer_webhook  
contexts:  
- context:  
  cluster: bouncer_webhook  
  user: api-server  
  name: bouncer_validator  
current-context: bouncer_validator  
preferences: {}  
users:  
- name: api-server  
  user:  
    client-certificate: /etc/kubernetes/pki/apiserver.crt  
    client-key: /etc/kubernetes/pki/apiserver.key
```

3. ImagePolicyWebhook Implementation



- Kube-APIServer level...

```
- --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook
- --admission-control-config-file
    =/etc/kubernetes/xyz/admission_configuration.json
```

4. Audit Policy

- YAML File to capture events of API Calls

```

apiVersion: audit.k8s.io/v1 # This is required.
kind: Policy
rules:
  # Log all ConfigMap and Secret changes at the Metadata level
  - level: Metadata
    resources:
      - group: ""
        resources: ["secrets", "configmaps"]
    namespaces: ["prod"]

apiVersion: audit.k8s.io/v1 # This is required.
kind: Policy
# Don't generate audit events for all requests
# in RequestReceived stage.
omitStages:
  - "RequestReceived"
rules:
  # Log pod changes at RequestResponse level
  - level: RequestResponse
    resources:
      - group: ""
        # Resource "pods" doesn't match requests
        # to any subresource of pods,
        # which is consistent with the RBAC policy.
        resources: ["pods"]

```

```

apiVersion: audit.k8s.io/v1 # This is required.
kind: Policy
rules:
  # Log the request body of configmap changes in kube-system.
  - level: Request
    verbs: ["get", "create", "update", "watch", "delete", "list"]
    resources:
      - group: "" # core API group
        resources: ["configmaps"]
    namespaces: ["kube-system"]

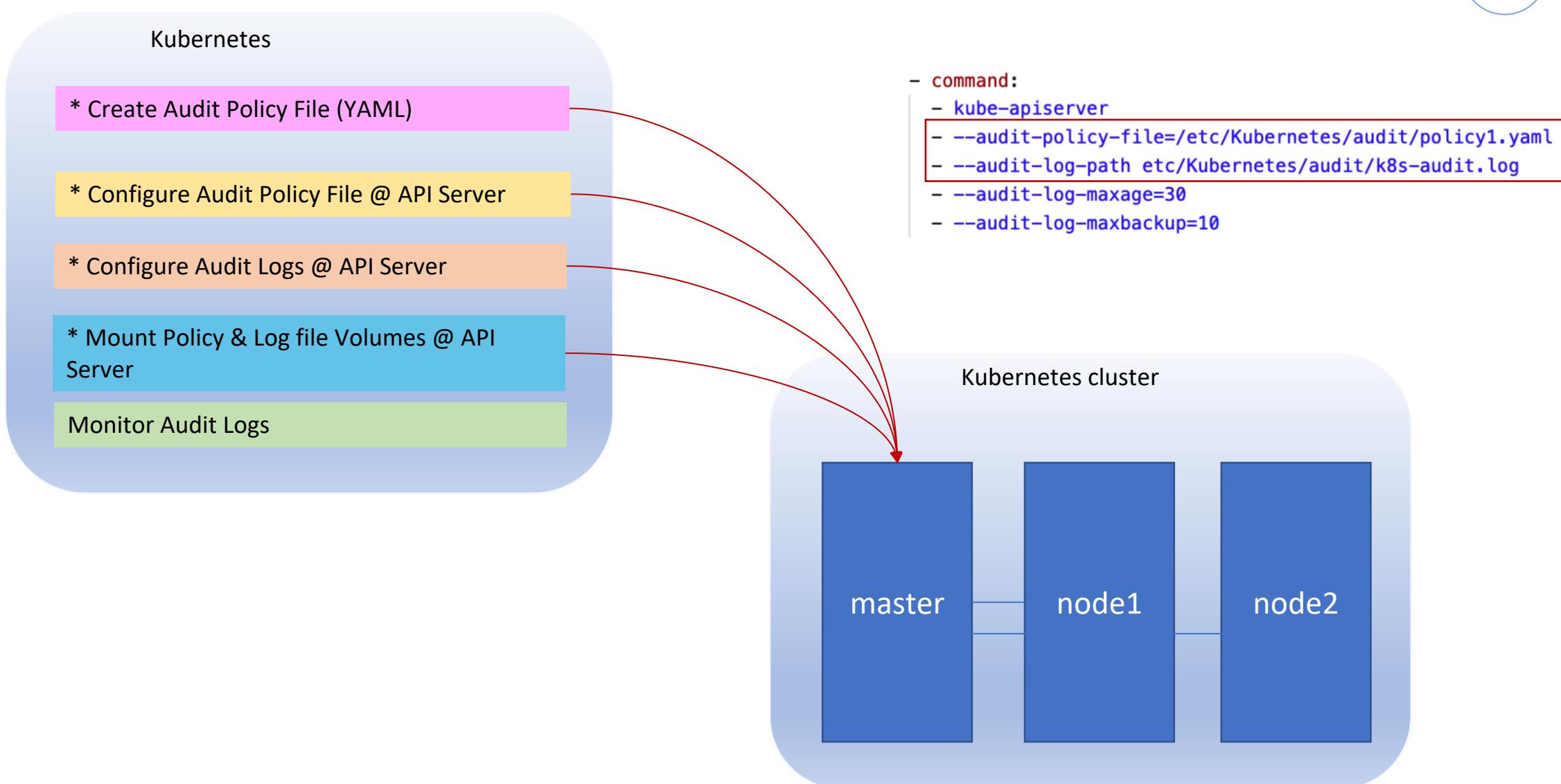
  # This rule only applies to resources in the "kube-system" namespace.
  # The empty string "" can be used to select non-namespaced resources.
  namespaces: ["kube-system"]

apiVersion: audit.k8s.io/v1 # This is required.
kind: Policy
rules:
  # Don't log requests to a configmap called "test-configmap"
  - level: None
    resources:
      - group: ""
        resources: ["configmaps"]
        resourceNames: ["test-configmap"]

  # Don't log watch requests by the "system:kube-proxy" endpoint
  - level: None
    users: ["system:kube-proxy"]
    verbs: ["watch"]
    resources:
      - group: "" # core API group
        resources: ["endpoints", "services"]

```

4. Audit Policy Implementation



5. Falco Rules

Falco Rule – YAML file

- **Default rules file**, installed at `/etc/falco/falco_rules.yaml`
- **Local rules file**, installed at `/etc/falco/falco_rules.local.yaml`
 - Update falco config `/etc/falco/falco.yaml` & restart falco

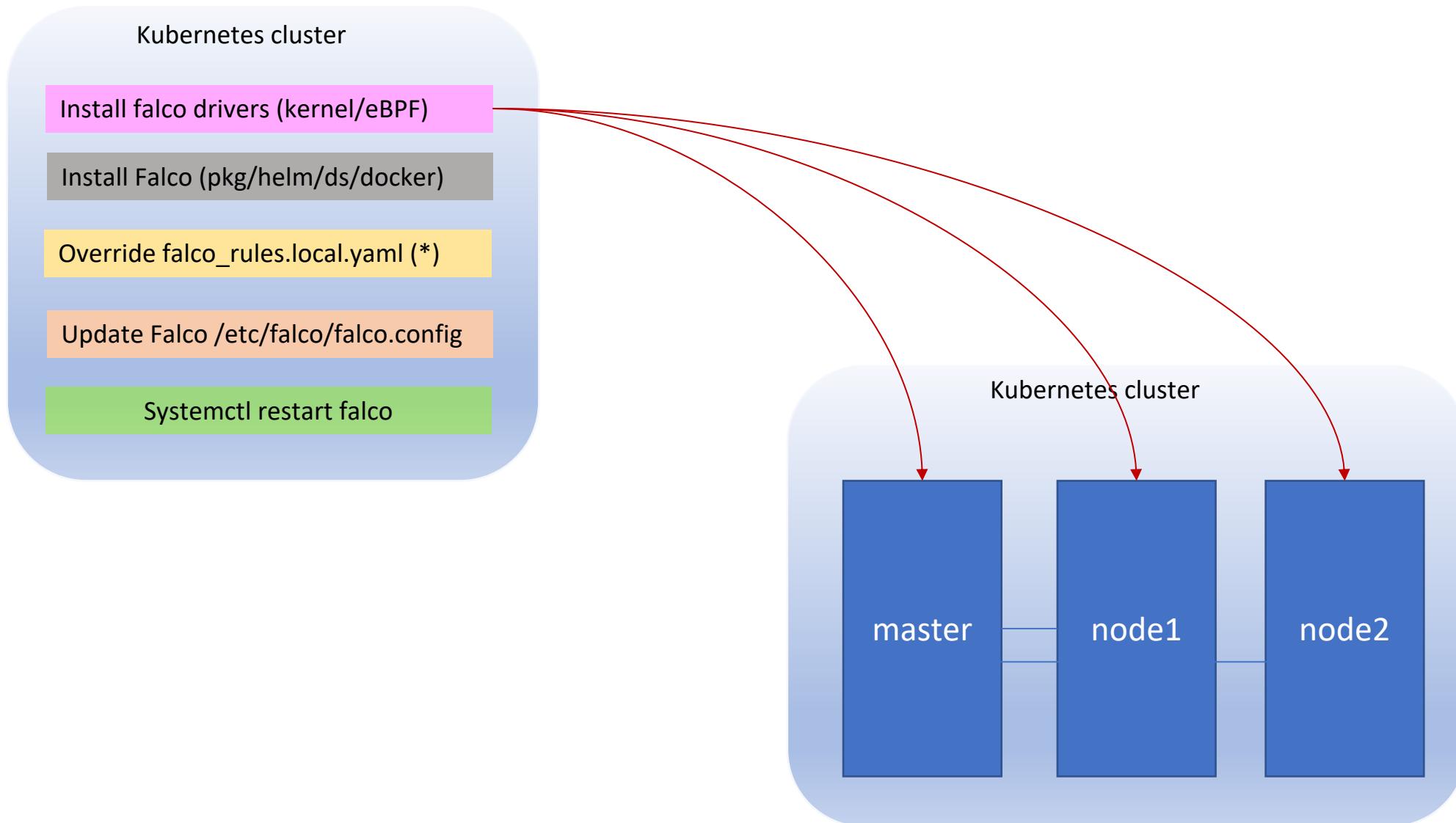
Priority: EMERGENCY/ALERT/CRITICAL/ERROR/WARNING/NOTICE/INFO/DEBUG

```
- macro: container
  condition: container.id != host

- macro: spawned_process
  condition: evt.type = execve and evt.dir=<

- rule: run_shell_in_container
  desc: a shell was spawned by a non-shell program in a container. Container entrypoints are excluded.
  condition: container and proc.name = bash and spawned_process and proc.pname exists and not proc.pname i
  output: "Shell spawned in a container other than entrypoint (user=%user.name container_id=%container.id"
  priority: WARNING
```

5. Falco Implementation



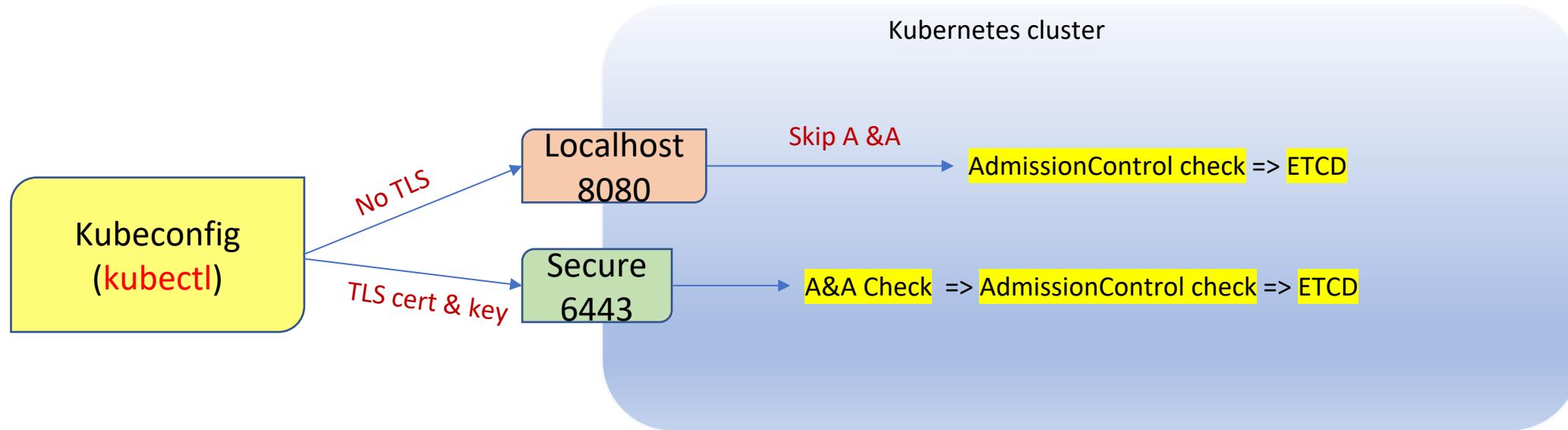
6. Tracing Containers

- `crictl ps -a`
- `crictl ps -id 123456789`
- `crictl inspect 123456789 | grep -i profile`
- `crictl inspect 123456789 | grep args -A2 -B2`

- `crictl pods --name pod1`
- `crictl pods -id 595af943c3245`
- `crictl pods | grep apparmor`
- `crictl ps --pod 21aacb8f4ca8d`
- `watch crictl ps`

- `ps aux | grep xyzprocess1.` (find PID)
- `strace -p 123` (find using strace)

7. Secure Kubernetes API Server



- **localhost** – default IP can be change with `--insecure-bind-address`
- **Secure** – default Port is **6443**, change with `--secure-port`
- **Secure** – set TLS certificate with `--tls-cert-file`
- **Secure** – set TLS certificate key with `--tls-private-key-file` flag

7. Disable Anonymous requests to API

- API Server level `--anonymous-auth=true` If its enabled...
- Other than API authentication methods are treated as anonymous requests
- ABAC and RBAC authorizers requires explicit
 - auth of `system:anonymous` user
 - `system:unauthenticated` group

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: anonymous-review-access
rules:
- apiGroups:
  - authorization.k8s.io
  resources:
  - selfsubjectaccessreviews
  - selfsubjectrulesreviews
  verbs:
  - create
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: anonymous-review-access
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: anonymous-review-access
subjects:
- kind: User
  name: system:anonymous
  namespace: default
```

8. Troubleshoot if API Server down

SSH Master Node...

```
cd /var/log/pods/
```

```
ls -l
```

```
cd kube-system_kube-apiserver-controlplane_xxx/kube-apiserver
```

```
tail -f 1.log
```

General Resource Related

1. NetworkPolicy
2. RBAC
3. Secret
4. ServiceAccount
5. AppArmor
6. SecComp
7. RuntimeClass
8. SecurityContext & Immutable Pods
9. Static Analysis
10. Trivy
11. Ingress TLS
12. OPA

1. Network Policy YAML

```

1  apiVersion: networking.k8s.io/v1
2  kind: NetworkPolicy
3  metadata:
4    name: test-network-policy
5    namespace: default
6  spec:
7    podSelector:
8      matchLabels:
9        role: db
10   policyTypes:
11     - Ingress
12     - Egress
13   ingress:
14     - from:
15       - ipBlock:
16         cidr: 172.17.0.0/16
17         except:
18           - 172.17.1.0/24
19       - namespaceSelector:
20         matchLabels:
21           project: myproject
22       - podSelector:
23         matchLabels:
24           role: frontend
25     ports:
26       - protocol: TCP
27         port: 6379
28   egress:
29     - to:
30       - ipBlock:
31         cidr: 10.0.0.0/24
32     ports:
33       - protocol: TCP
34         port: 5978

```

Name of the NetworkPolicy & applicable Domain

Target Pod: To **which Pod** you want to apply this ?

What type of Policy ? Incoming & Outgoing of above Pod

From where you want to allow? If its blank then Deny all

From which CIDR you want to allow?

Except which CIDR (filter) ?

From what Namespace you want to allow ?

From which Pod you want to allow traffic?

Incoming Policy on what Port of Pod?

To where you want to allow outgoing? If its blank then Deny all

To which CIDR you want to allow outgoing?

Outgoing Policy on what Port of Pod?

IMPORTANT: if YAML is specified as...
 > List then OR condition
 > If not then AND condition

2. RBAC

```

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: configmap-updater
rules:
- apiGroups: [""]
  #
  # at the HTTP level, the name of the resource objects is "configmaps"
  resources: ["configmaps"]
  resourceNames: ["my-configmap"]
  verbs: ["update", "get"]

```

```

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  # "namespace" omitted since ClusterRoles are not namespaced
  name: secret-reader
rules:
- apiGroups: [""]
  #
  # at the HTTP level, the name of the resource for accessing Secret
  # objects is "secrets"
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]

```

```

apiVersion: rbac.authorization.k8s.io/v1
# This role binding allows "jane" to read pods
# You need to already have a Role named "pod-reader"
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
# You can specify more than one "subject"
- kind: User
  name: jane # "name" is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
# "roleRef" specifies the binding to a Role
kind: Role #this must be Role or ClusterRole
name: pod-reader # this must match the name of the Role
apiGroup: rbac.authorization.k8s.io

```

```

apiVersion: rbac.authorization.k8s.io/v1
# This cluster role binding allows anyone in the "manager" group
# to read secrets
kind: ClusterRoleBinding
metadata:
  name: read-secrets-global
subjects:
- kind: Group
  name: manager # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io

```

RBAC Commands

```
Kubectl create ns test
```

```
Kubectl create sa svc-user1 -n test
```

```
Kubectl create role creator-role -n test --  
verb=create --resource=secret --resource=configmap
```

```
Kubectl create rolebinding creator-binding -n test  
--role creator-role --serviceaccount test:svc-  
user1
```

```
Kubectl auth can-i create secret -n test --as  
system:serviceaccount:test:svc-user1
```

3. Secrets as environment variables

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  USER_NAME: YWRtaW4=
  PASSWORD: MjAyMzEwMjMxNjUy
```

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
  - name: mycontainer
    image: redis
    env:
      - name: SECRET_USERNAME
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: username
      - name: SECRET_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: password
  restartPolicy: Never
```

3. Secrets as Volumemount

```
kubectl create secret generic ssh-key-secret --from-file=ssh-privatekey=/path/to/.ssh/id_rsa --from-file=ssh-publickey=/path/to/.ssh/id_rsa.pub
```

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-test-pod
  labels:
    name: secret-test
spec:
  volumes:
  - name: secret-volume
    secret:
      secretName: ssh-key-secret
  containers:
  - name: ssh-test-container
    image: mySshImage
    volumeMounts:
    - name: secret-volume
      readOnly: true
      mountPath: "/etc/secret-volume"
```

/etc/secret-volume/ssh-publickey
/etc/secret-volume/ssh-privatekey

3. Decode Secret using base64

```
kubectl create secret generic db-user-pass \  
  --from-literal=username=devuser \  
  --from-literal=password='S!B\*d$zDsb='
```

```
kubectl describe secrets/db-user-pass
```

```
Name:          db-user-pass  
Namespace:    default  
Labels:        <none>  
Annotations:   <none>  
  
Type:         Opaque  
  
Data
```

```
=====  
password:    12 bytes  
username:    5 bytes
```

```
kubectl get secret db-user-pass -o jsonpath='{.data}'  
  
{"password":"MwYyZDFlMmU2N2Rm", "username":"YWRtaW4="}
```

```
echo 'MwYyZDFlMmU2N2Rm' | base64 --decode
```

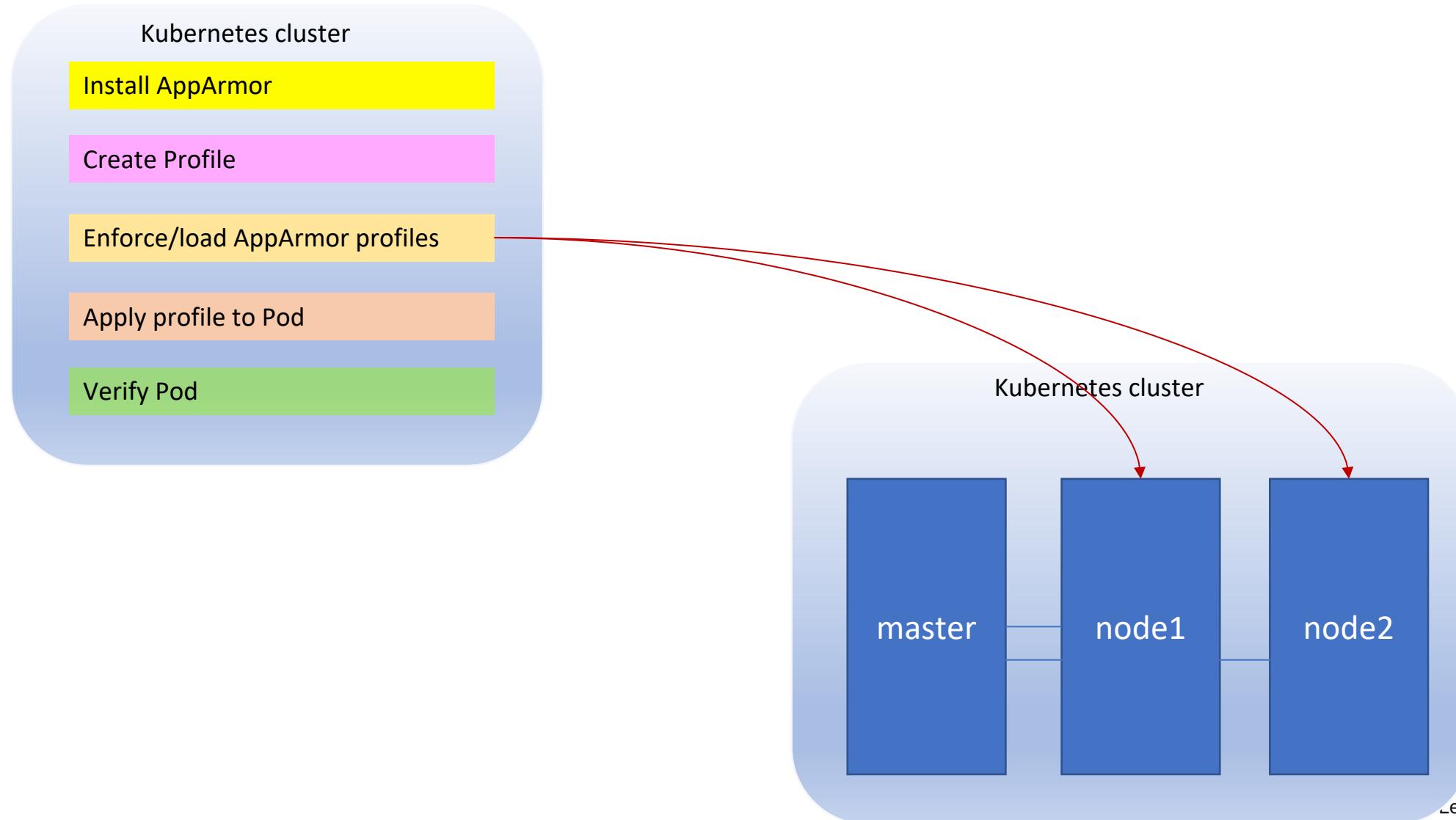
```
1f2d1e2e67df
```

4. ServiceAccount AutoMount

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: test-svc
  namespace: dev
automountServiceAccountToken: false
```

```
apiVersion: v1
kind: Pod
metadata:
  name: web
  namespace: dev
spec:
  containers:
  - image: web
    name: web
  serviceAccountName: test-svc
  automountServiceAccountToken: false
```

5. AppArmor Implementation



5. AppArmor Profile & Apply to Pod

```
#include <tunables/global>

profile k8s-apparmor-example-deny-write flags=(attach_disconnected) {
    #include <abstractions/base>

    file,
    # Deny all file writes.
    deny /** w,
}
```

```
apparmor_parser /given/path/file
aa-status | grep profile_name
```

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-apparmor-2
annotations:
  container.apparmor.security.beta.kubernetes.io/hello: localhost/k8s-apparmor-example-allow-write
spec:
  containers:
  - name: hello
    image: busybox
    command: [ "sh", "-c", "echo 'Hello AppArmor!' && sleep 1h" ]
```

6. SecComp Profiles

audit.json violation.json fine-grained.json

```
{
  "defaultAction": "SCMP_ACT_ERRNO",
  "architectures": [
    "SCMP_ARCH_X86_64",
    "SCMP_ARCH_X86",
    "SCMP_ARCH_X32"
  ],
  "syscalls": [
    {
      "names": [
        "accept4",
        "epoll_wait",
        "recvfrom",
        "sendto",
        "set_tid_address",
        "setitimer",
        "writev"
      ],
      "action": "SCMP_ACT_ALLOW"
    }
  ]
}
```

```

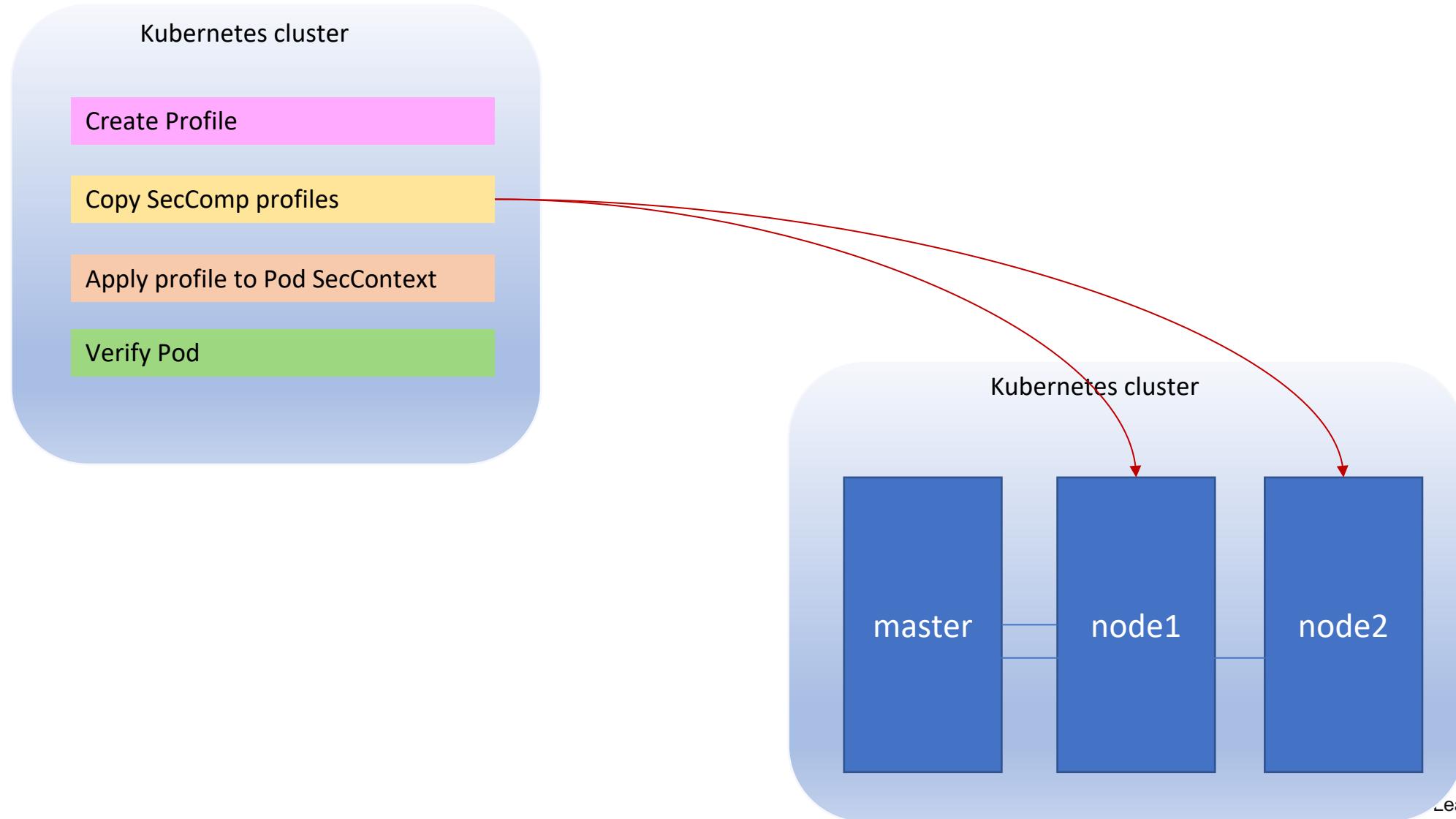
apiVersion: v1
kind: Pod
metadata:
  name: fine-pod
  labels:
    app: fine-pod
spec:
  securityContext:
    seccompProfile:
      type: Localhost
      localhostProfile: profiles/fine-grained.json
  containers:
    - name: test-container
      image: hashicorp/http-echo:0.2.3
      args:
        - "-text=just made some syscalls!"
    securityContext:
      allowPrivilegeEscalation: false

```

Profiles should be placed at

/var/lib/kubelet/seccomp/profiles/fine-grained.json

6. SecComp Implementation



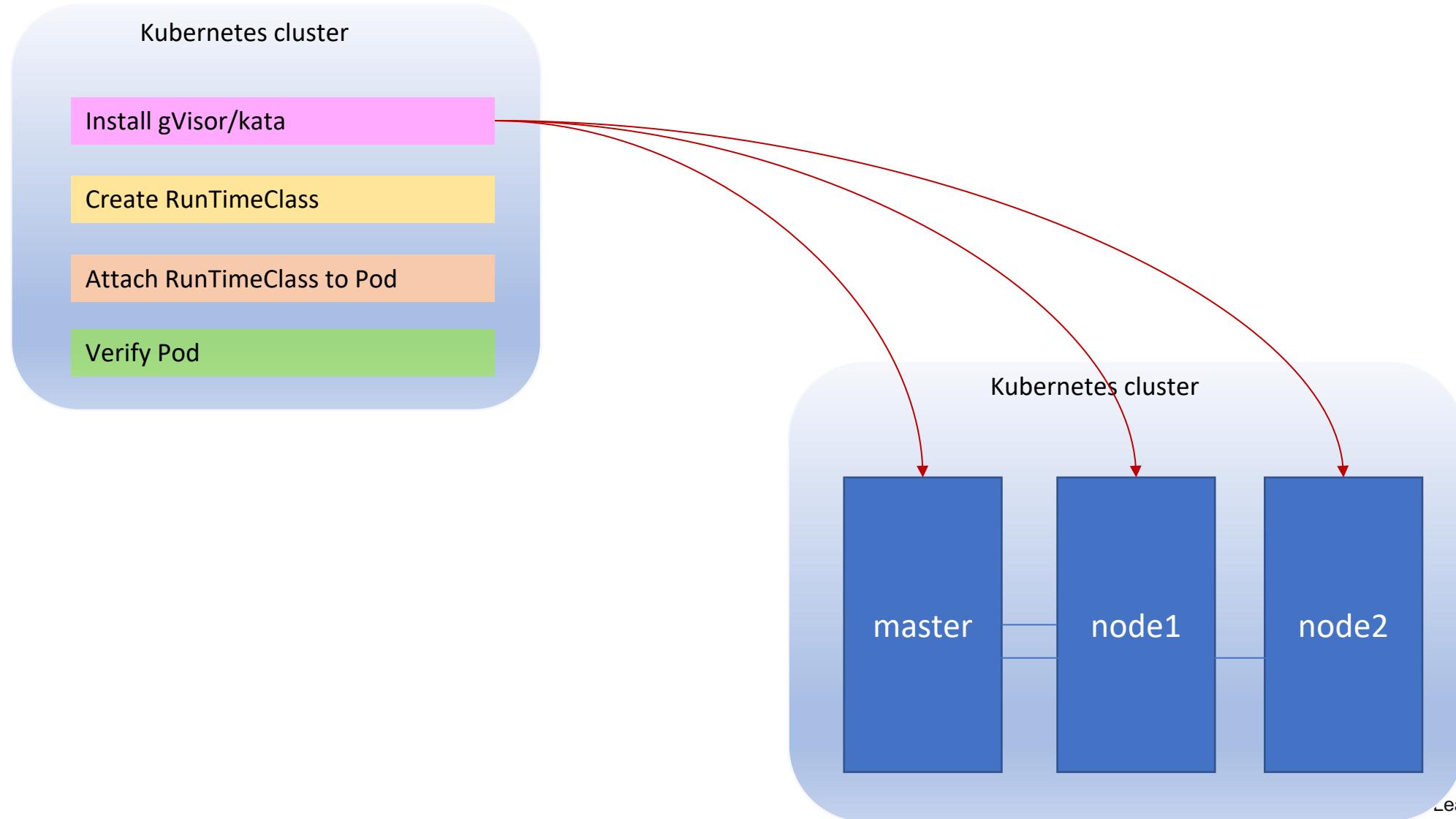
7. RuntimeClass

```
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: myclass # The name
handler: runsc # non-namespaced
```

```
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: myclass # The name
handler: kata # non-namespaced
```

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  runtimeClassName: myclass
  containers:
    - image: busybox
      name: busybox
      command: ['sh', '-c', 'while true; do echo
```

7. RuntimeClass Implementation



8. SecurityContext @ Pod level

```
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
```

Security Context @ Container Level

```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo-2
spec:
  securityContext:
    runAsUser: 1000
  containers:
  - name: sec-ctx-demo-2
    image: gcr.io/google-samples/node-hello:1.0
    securityContext:
      runAsUser: 2000
      allowPrivilegeEscalation: false
```

```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo-4
spec:
  containers:
  - name: sec-ctx-4
    image: gcr.io/google-samples/node-hello:1.0
    securityContext:
      capabilities:
        add: ["NET_ADMIN", "SYS_TIME"]
```

Seccomp/seLinux for a Container

```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo-4
spec:
  containers:
    - name: sec-ctx-4
      image: gcr.io/google-samples/node-hello:1.0
      securityContext:
        capabilities:
          add: ["NET_ADMIN", "SYS_TIME"]
```

```
...
  securityContext:
    seccompProfile:
      type: RuntimeDefault
```

```
...
  securityContext:
    seccompProfile:
      type: Localhost
      localhostProfile: my-profiles/profile-allow.json
```

```
...
  securityContext:
    seLinuxOptions:
      level: "s0:c123,c456"
```

8.2. Immutable Pods

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: non-legitimate-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      app: non-legitimate-deploy
  template:
    metadata:
      labels:
        app: non-legitimate-deploy
    spec:
      hostIPC: true
      hostNetwork: true
      hostPID: true
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - containerPort: 80
      securityContext:
        privileged: true
        runAsUser: 0
        allowPrivilegeEscalation: true
        readOnlyRootFilesystem: true
        capabilities:
          add: ["CAP_SYS_BOOT"]

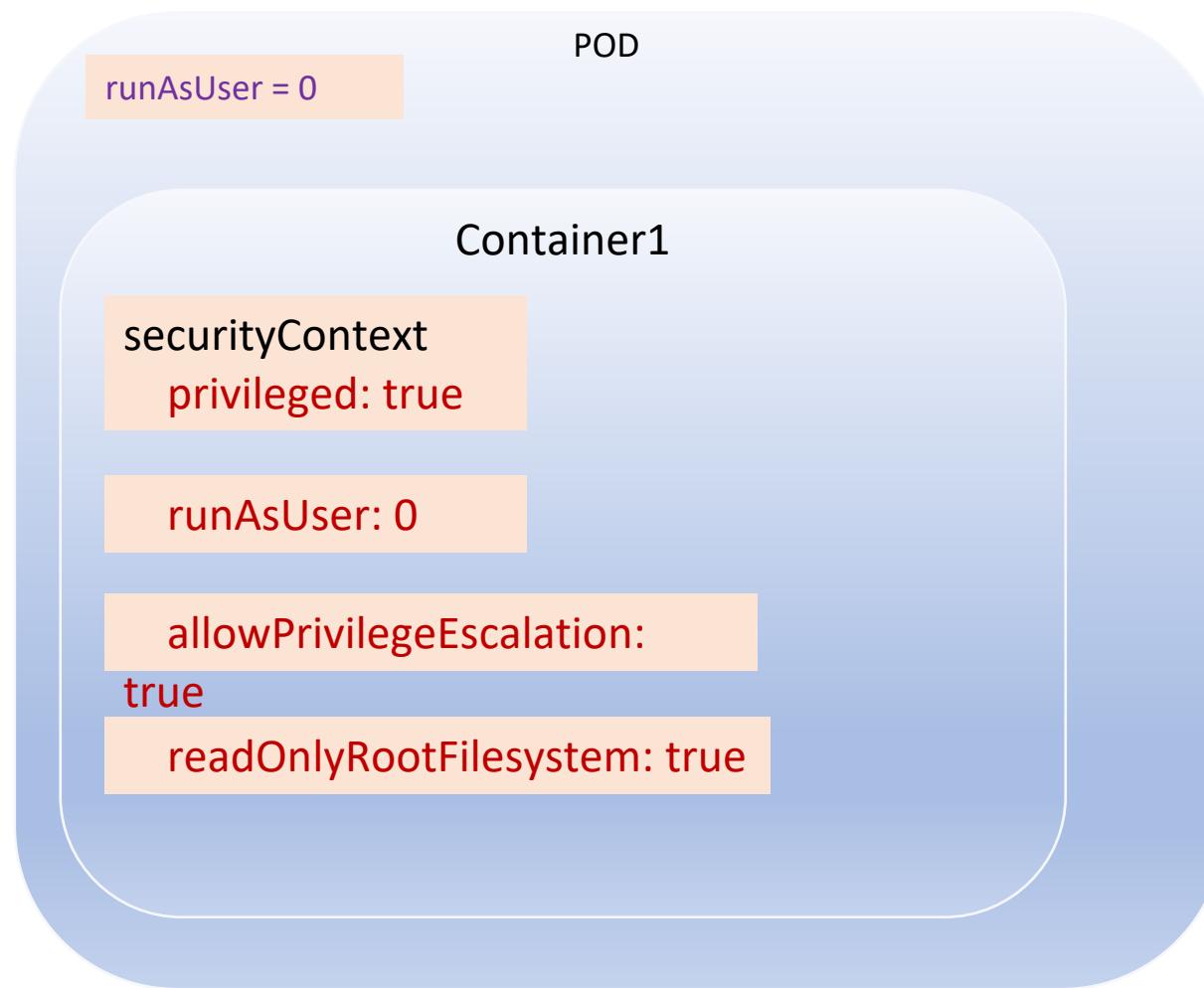
```

```

apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo
spec:
  volumes:
    - name: sec-ctx-vol
      emptyDir: {}
  containers:
    - name: sec-ctx-demo
      image: busybox
      command: [ "sh", "-c", "echo hello > /data/demo/hello.txt; sleep 1h" ]
  securityContext:
    privileged: true
    runAsUser: 0
    allowPrivilegeEscalation: true
    readOnlyRootFilesystem: false
  volumeMounts:
    - name: sec-ctx-vol
      mountPath: /data/demo

```

Immutability - Avoid Elevated Privileges



9. Static Analysis (Manual)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: non-legitimate-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      app: non-legitimate-deploy
  template:
    metadata:
      labels:
        app: non-legitimate-deploy
    spec:
      hostIPC: true
      hostNetwork: true
      hostPID: true
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - containerPort: 80
          securityContext:
            privileged: true
            runAsUser: 0
            allowPrivilegeEscalation: true
            readOnlyRootFilesystem: true
            capabilities:
              add: ["CAP_SYS_BOOT"]
FROM fedora:29
RUN groupadd -r swuser -g 433 && \
    useradd -u 431 -r -g swuser -s /sbin/nologin -c "Docker image user" swuser
USER root
RUN dnf install -y vim
USER swuser
ENV USER_NAME="bob"
ENV PASSWORD="abcd123"
```

10. Trivy Scan



```
trivy image -s HIGH,CRITICAL python:3.4-alpine | grep -i total
```

```
Total: 37 (UNKNOWN: 0, LOW: 4, MEDIUM: 16, HIGH: 13, CRITICAL: 4)
```

Practice ... Practice... Practice...

All the Best & Good Luck