Learn with GVR

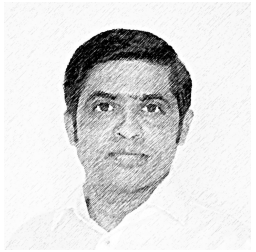
CKS Exam

Experience

Topics && Tips

Who Iam I

Learn with GVR



Venkata Ramana Gali

- *Senior Cloud DevOps Engineer/Architect*

LinkedIn https://www.linkedin.com/in/venkataramanagali

YouTube Learn with GVR

https://www.youtube.com/channel/UCwopwnnBoKMOUEOl6lefM0w/videos

GitHub https://github.com/ramanagali

CKS Exam

Learn with GVR

- 100 Marks
- 2 Hours (No Count down timer but Progress Indicator)
- 16 Questions (between 15 - 20)
- v1.23
- Results in 24hours
- 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

Learn with GVR

- **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

Learn with GVR

- **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

Learn with GVR

- 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

Learn with GVR

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

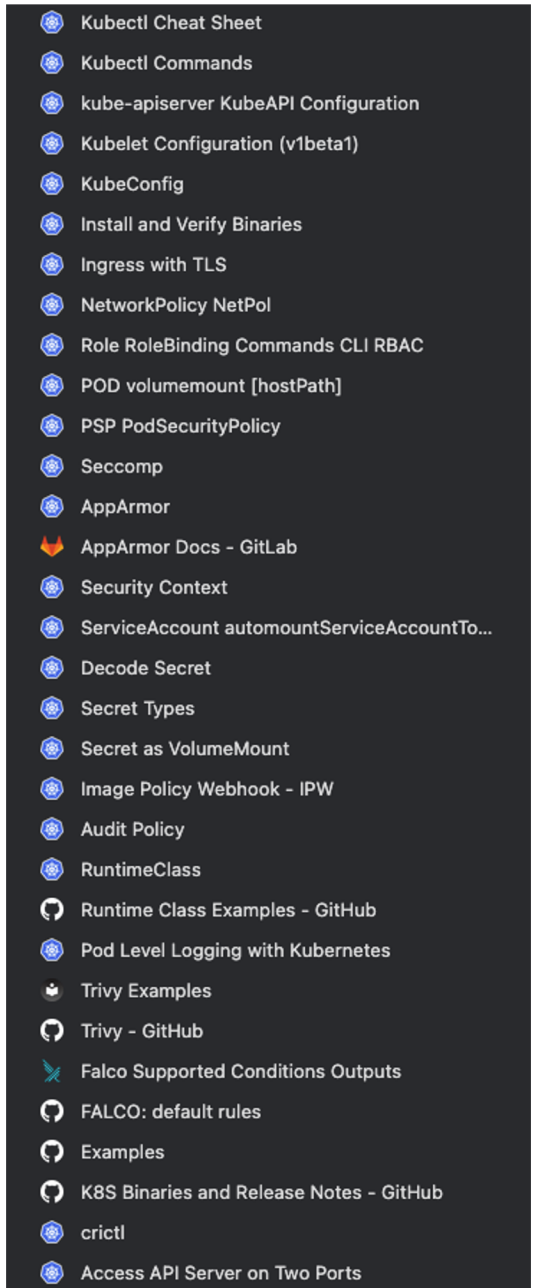
During the exam

Learn with GVR

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

Use Documentation BookMarks

Learn with GVR



CKS Exam Categories

Learn with GVR

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

Control Plane related

Learn with GVR

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

Learn with GVR

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

Learn with GVR

- **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

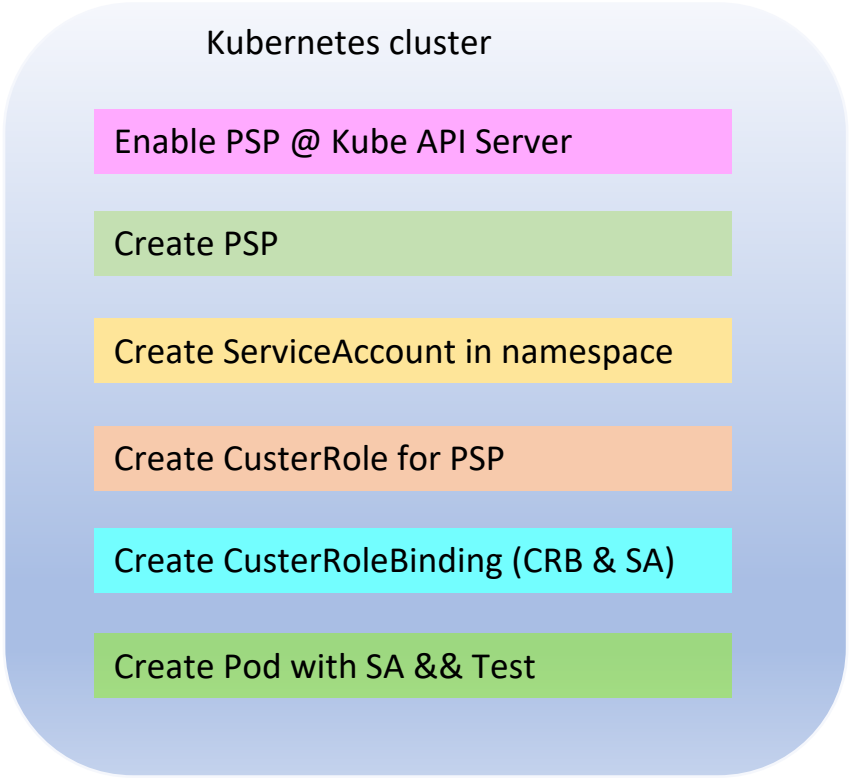
Learn with GVR

```yaml
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:
  - '*'
```

```yaml
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

Learn with GVR



Kube-APIServer level…

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

3. ImagePolicyWebhook

Learn with GVR

/etc/kubernetes/xyz/admission_kube_config.json

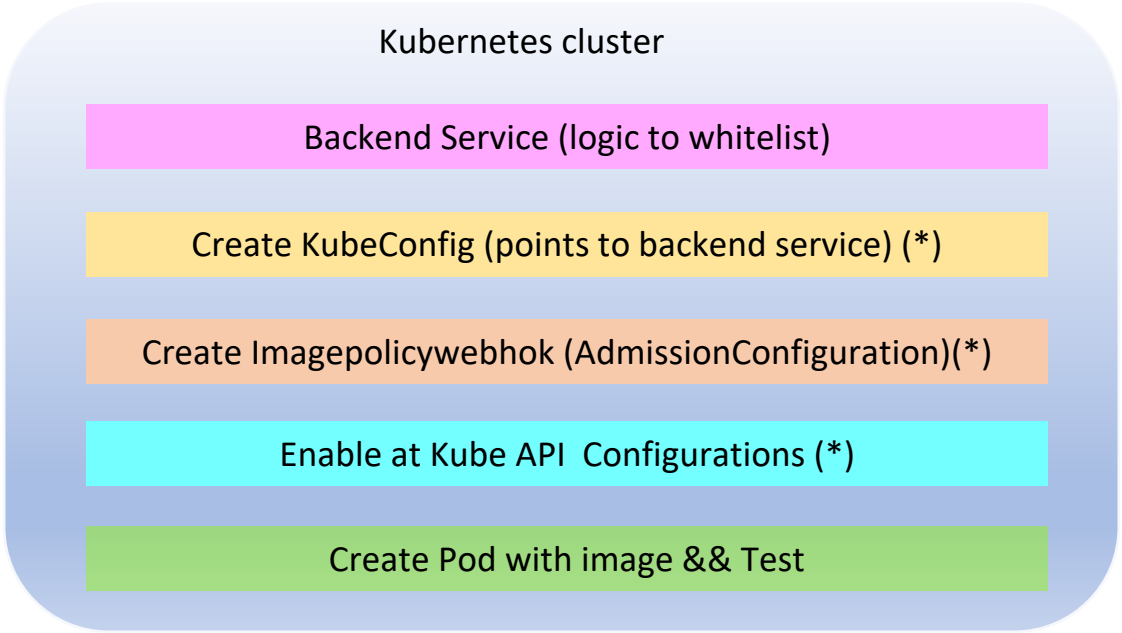
```
{
  "imagePolicy": {
    "kubeConfigFile": "/etc/kubernetes/pki/webhook/admission_kube_config.yaml",
    "allowTTL": 50,
    "denyTTL": 50,
    "retryBackoff": 500,
    "defaultAllow": false
  }
}
```

```
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

Learn with GVR



- Kube-APIServer level…

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

4. Audit Policy

Learn with GVR

- **YAML File** to capture events of API Calls

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

```yaml
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"]
    # 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"]
```

```yaml
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"]
```

```yaml
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" on end
  - level: None
    users: ["system:kube-proxy"]
    verbs: ["watch"]
    resources:
      - group: "" # core API group
        resources: ["endpoints", "services"]
```

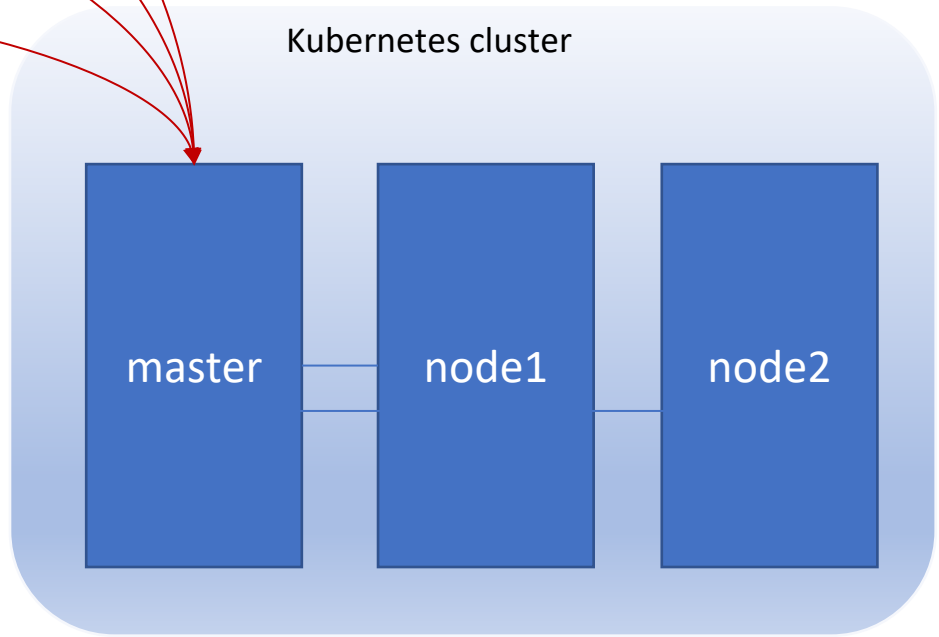
4. Audit Policy Implementation

Learn with GVR

Kubernetes

* Create Audit Policy File (YAML)
* Configure Audit Policy File @ API Server
* Configure Audit Logs @ API Server
* Mount Policy & Log file Volumes @ API Server
Monitor Audit Logs

```
- command:
  - kube-apiserver
  - --audit-policy-file=/etc/Kubernetes/audit/policy1.yaml
  - --audit-log-path etc/Kubernetes/audit/k8s-audit.log
  - --audit-log-maxage=30
  - --audit-log-maxbackup=10
```



5. Falco Rules

Learn with GVR

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

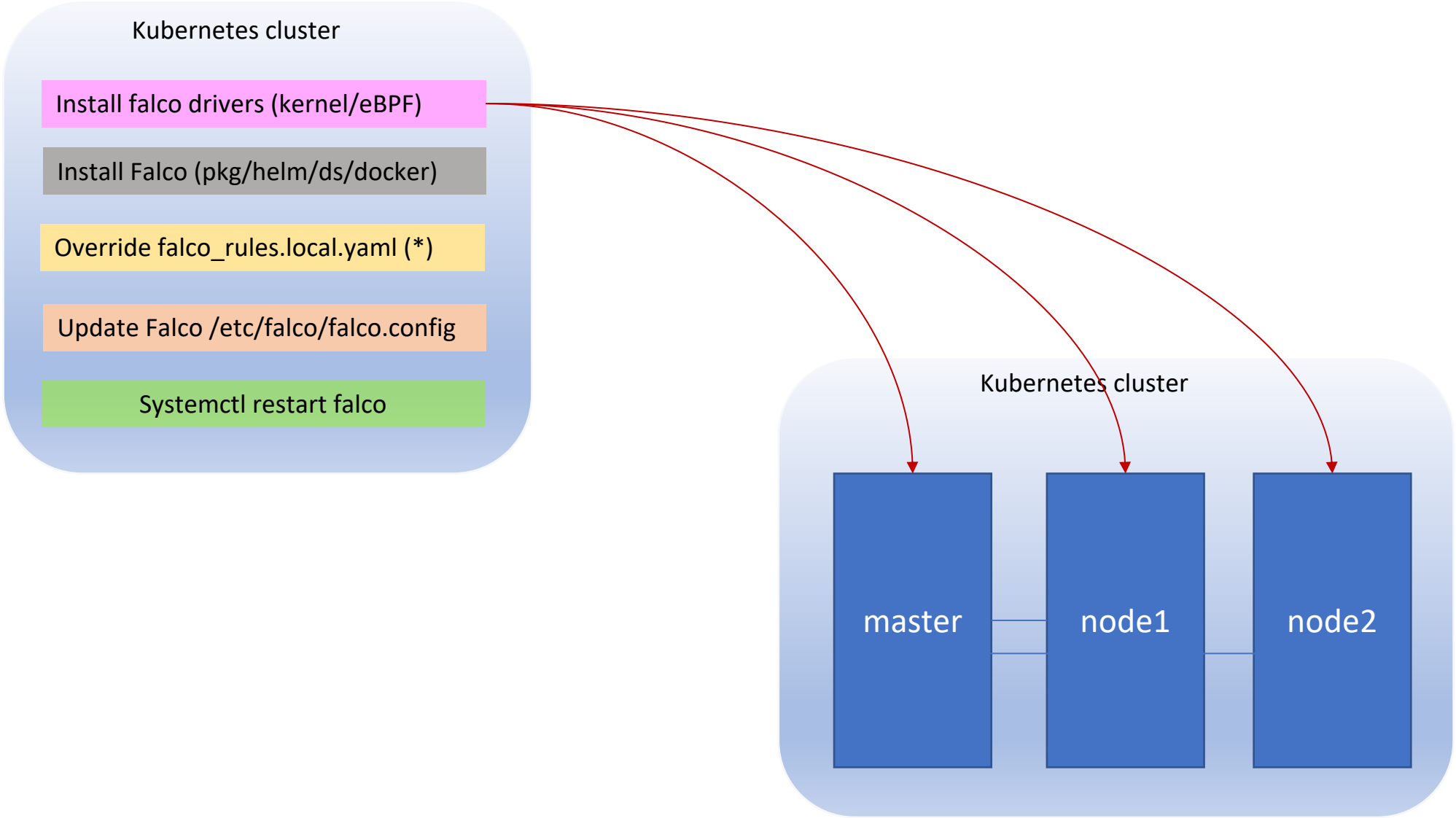
Priority: **EMERGENCY/ALERT/CRITICAL/ERROR/WARNING/NOTICE/INFO/DEBUG**

```
- 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
```

/R

5. Falco Implementation

Learn with GVR



6. Tracing Containers

Learn with GVR

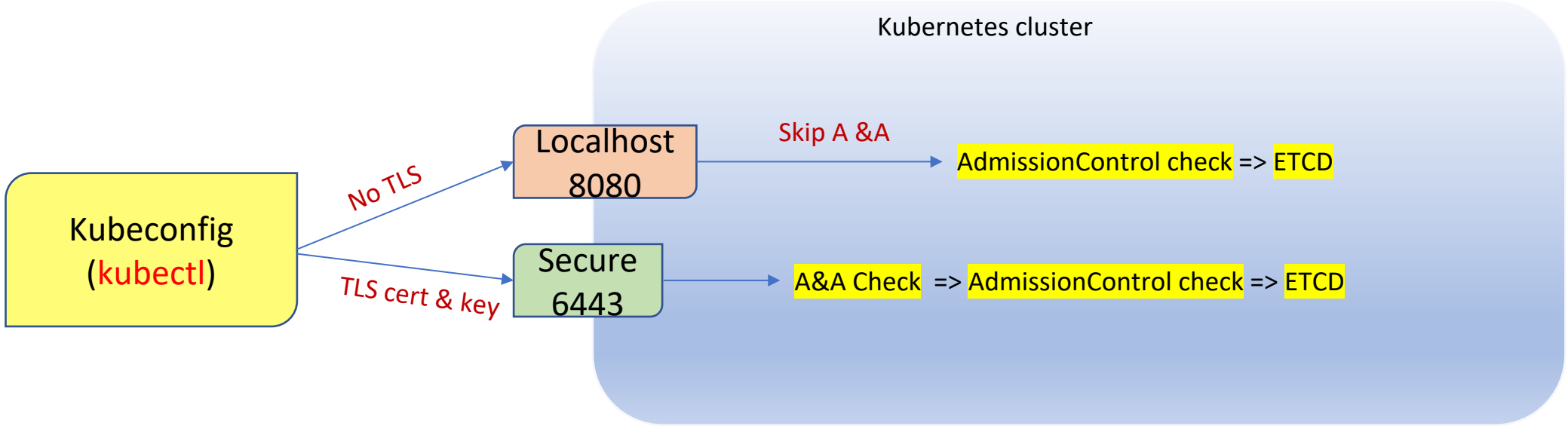
- `crictl ps -a`
- `crictl ps -id 123456789`
- `crictl inspect 123456789 | grep -i profile`
- `crictl inspect 123456789 | grep args –A2 –B2`

- `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

Learn with GVR



- **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

Learn with GVR

- 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

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

```yaml
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

Learn with GVR

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

Learn with GVR

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

Learn with GVR

```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
```

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?

IMPORTENT: if YAML is specified as…
> List then OR condition
> If not then AND condition

2. RBAC

Learn with GVR

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

```yaml
apiVersion: rbac.authorization.k8s.io/v1
# This role binding allows "jane" to read
# You need to already have a Role named "p
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 R
  kind: Role #this must be Role or ClusterR
  name: pod-reader # this must match the na
  apiGroup: rbac.authorization.k8s.io
```

```yaml
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"]
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
# This cluster role binding allows anyone
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

Learn with GVR

```
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

Learn with GVR

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  USER_NAME: YWRtaW4=
  PASSWORD: MWYyZDFlMmU2N2Rm
```

```yaml
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

Learn with GVR

```
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
```

```yaml
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

Learn with GVR

```
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

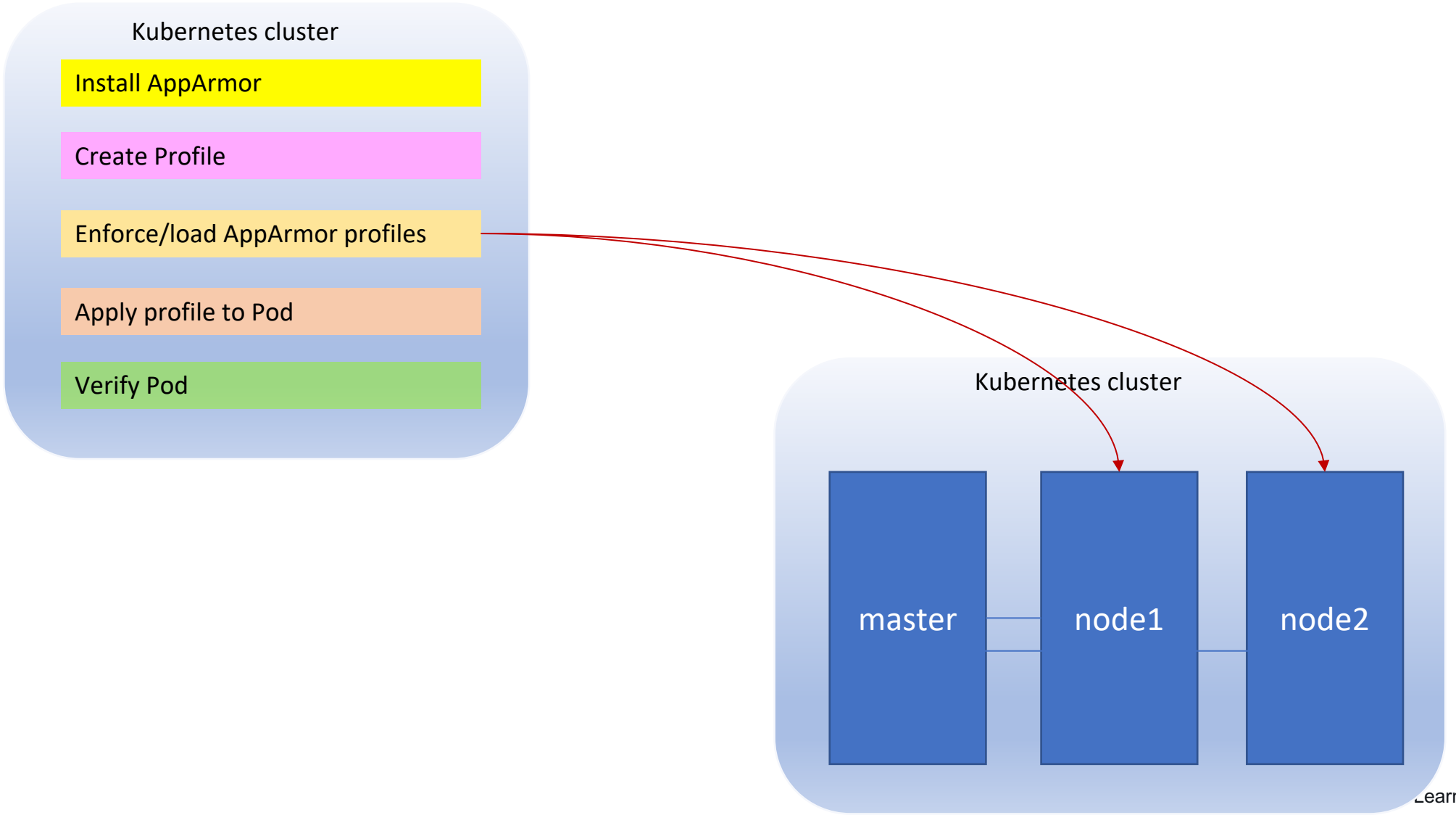
Learn with GVR

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

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

5. AppArmor Implementation

Learn with GVR



5. AppArmor Profile & Apply to Pod

Learn with GVR

```
#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

Learn with GVR

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"
        }
    ]
}
```

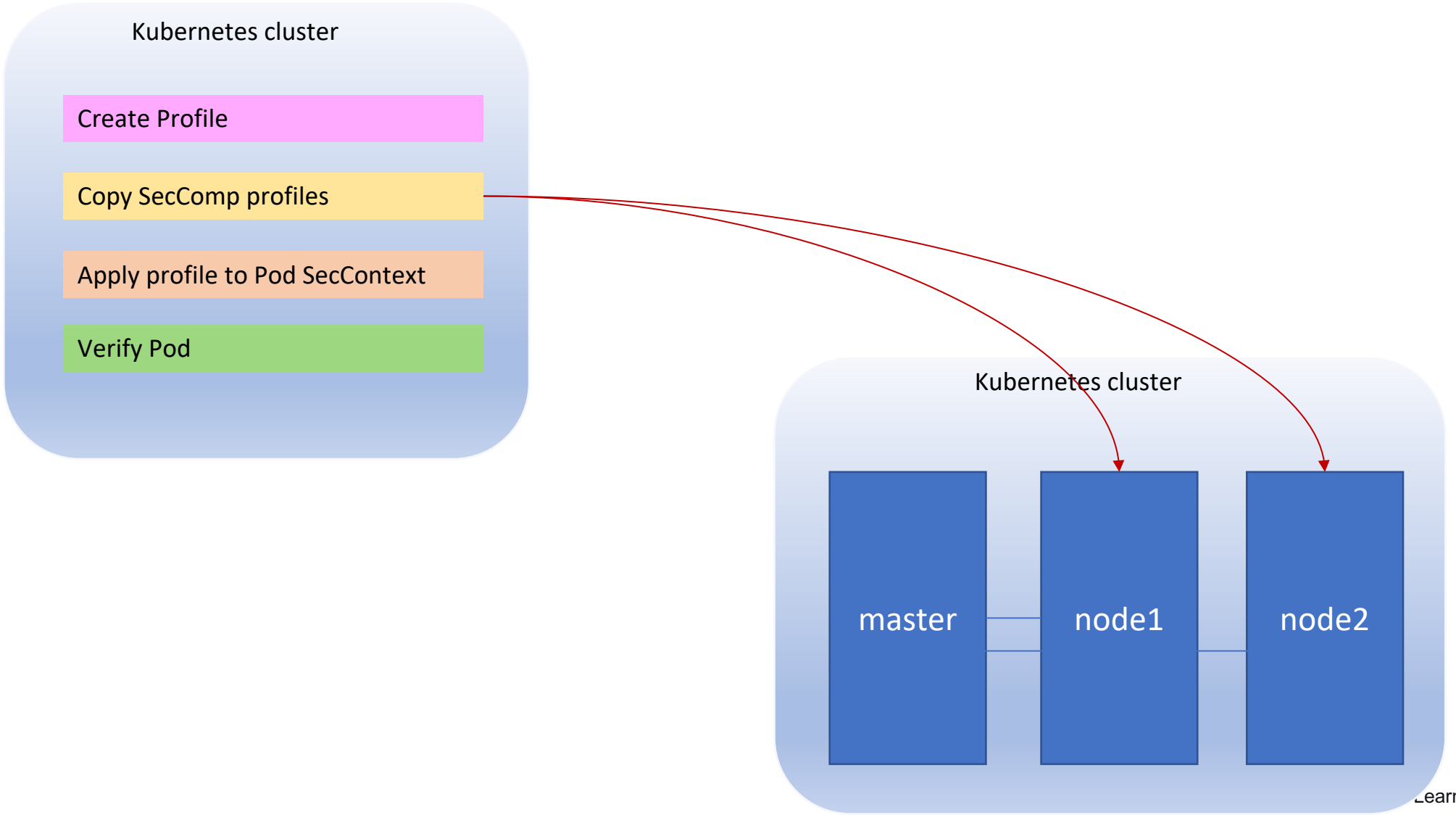
```
apiVersion: v1
kind: Pod
metadata:
  name: fine-pod
  labels:
    app: fine-pod
spec:
  securityContext:
    seccompProfile:
      type: Localhost
      localhostProfile: profiles/fine-grained.json
  containers:
  - name: test-container
    image: hashicorp/http-echo:0.2.3
    args:
    - "-text=just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

Profiles should be placed at

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

6. SecComp Implementation

Learn with GVR



7. RuntimeClass

Learn with GVR

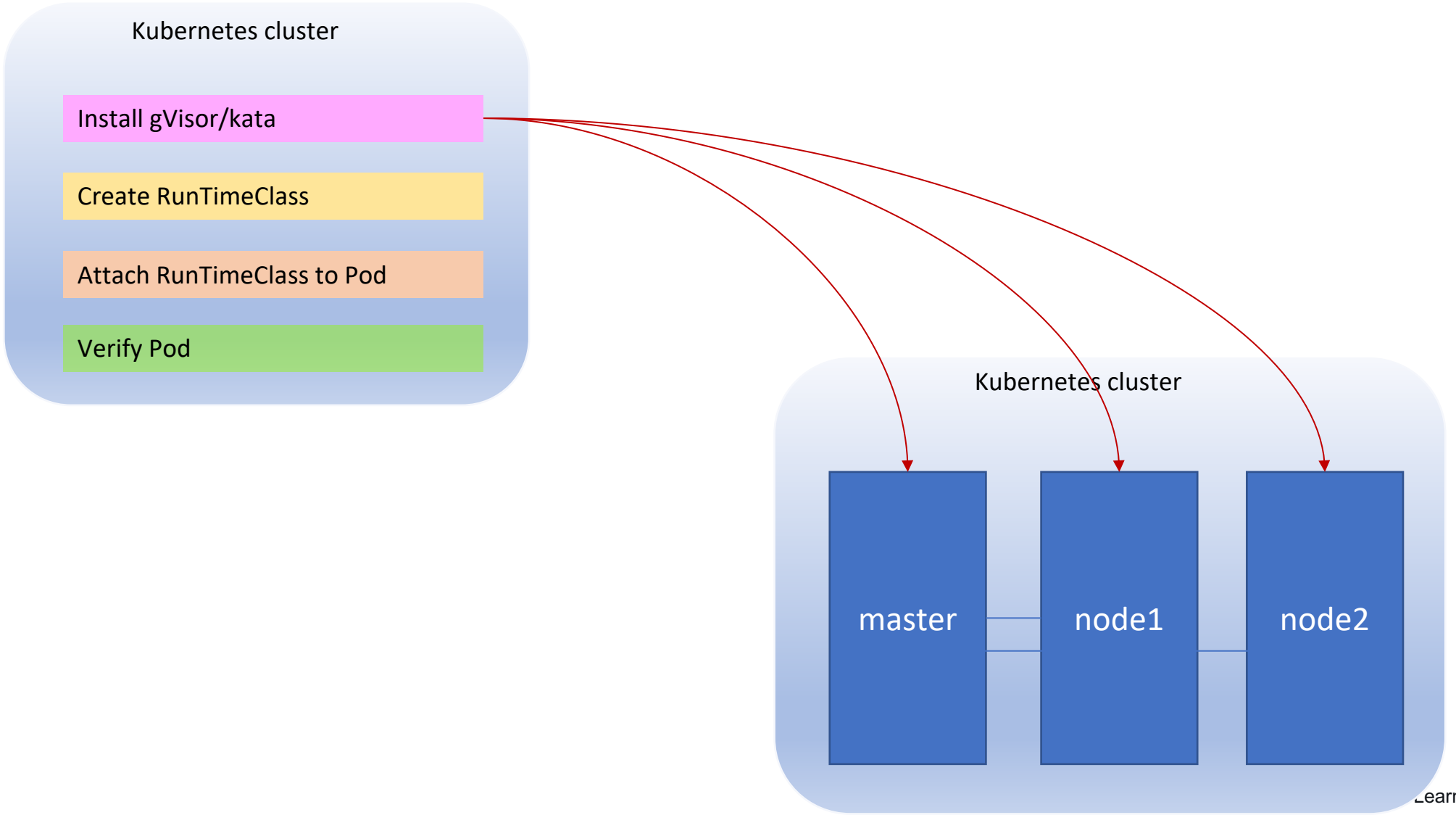
```yaml
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: myclass  # The name
handler: runsc  # non-namesp
```

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

```yaml
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

Learn with GVR



8. SecurityContext @ Pod level

Learn with GVR

```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
```

Security Context @ Container Level

Learn with GVR

```yaml
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
```

```yaml
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

Learn with GVR

```yaml
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"]
```

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

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

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

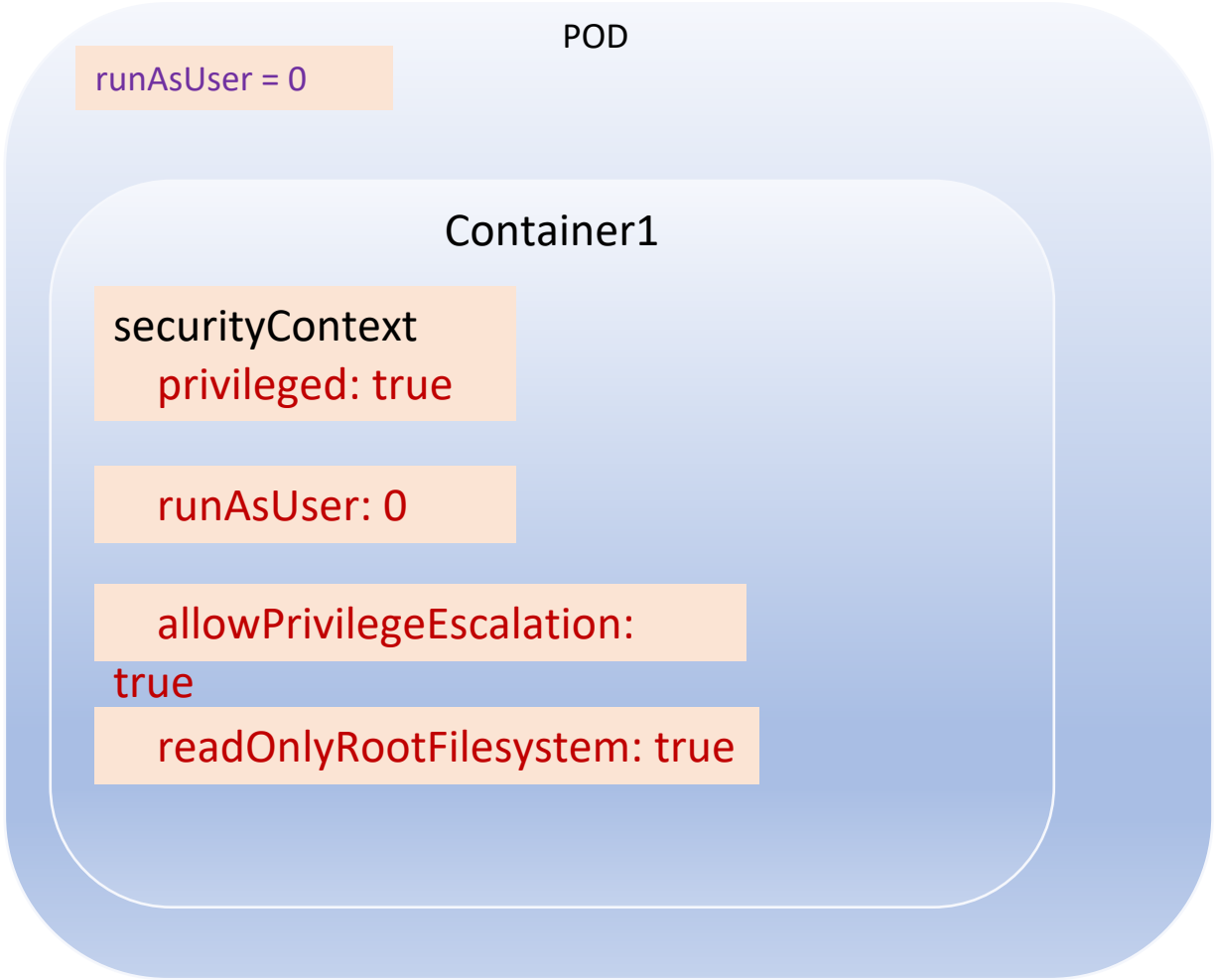
8.2. Immutable Pods

Learn with GVR

```yaml
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"]
```

```yaml
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)

Learn with GVR

```yaml
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

Learn with GVR

aqua trivy

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

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

Learn with GVR

Practice … Practice… Practice…

Learn with GVR

All the Best & Good Luck