Certified Kubernetes Security Specialist

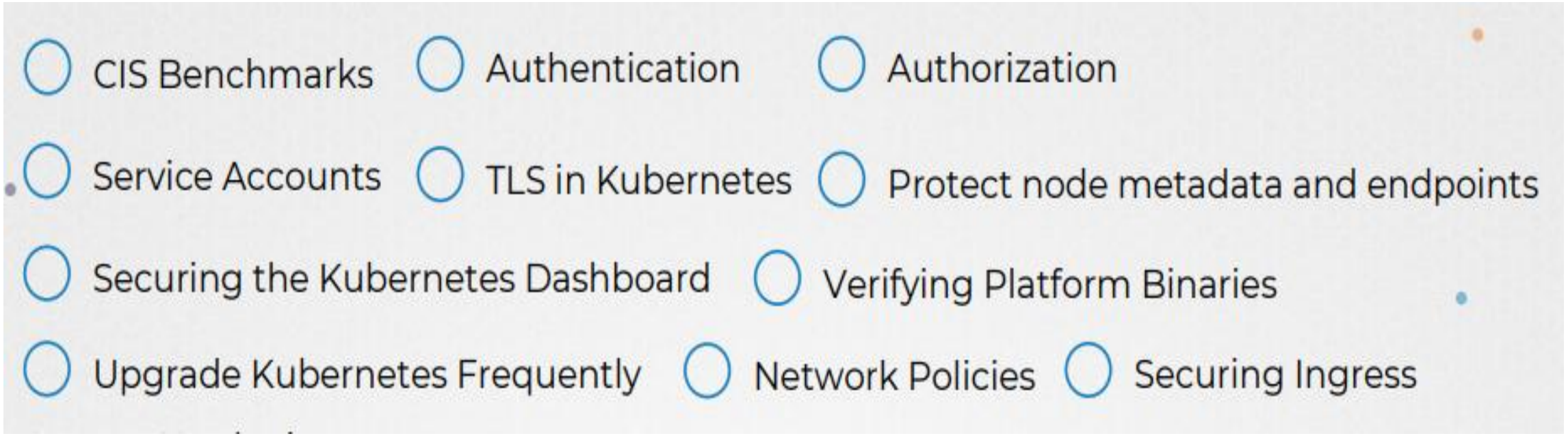
Nobelprog

Course Objectives

- Kubernetes Architecture review
- Understanding Kubernetes Attack Surface
- Cluster Setup and Hardening
- System Hardening
- Minimize Microservice Vulnerability
- Supply Chain Security
- Monitoring, Logging and Runtime Security

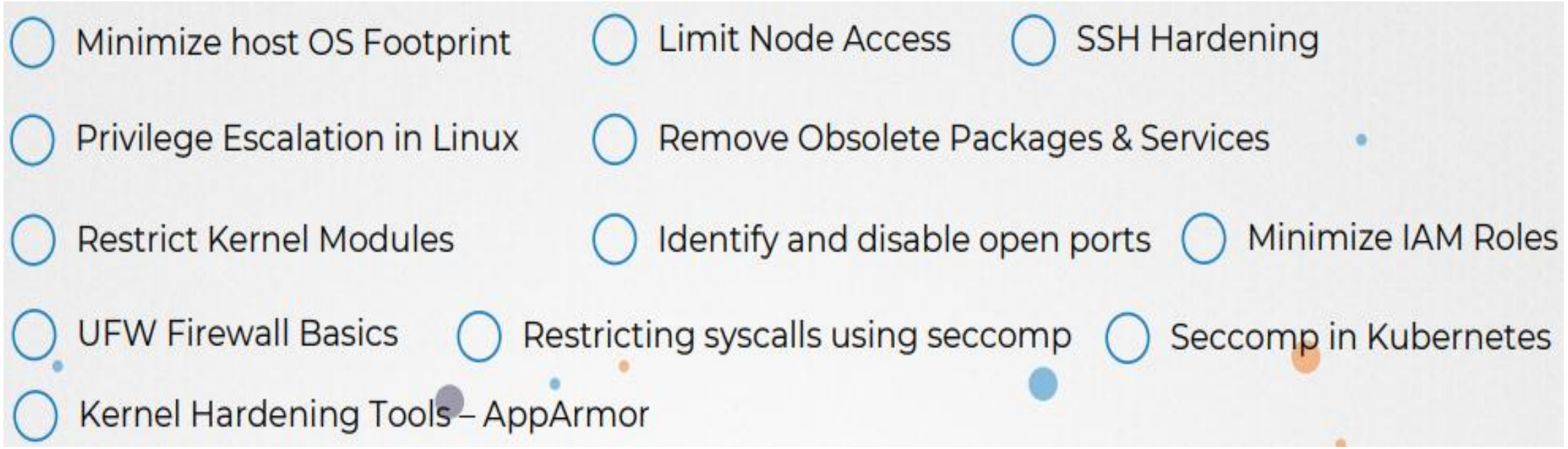
Course Objectives

- Cluster Setup and Hardening



Course Objectives

- System Hardening

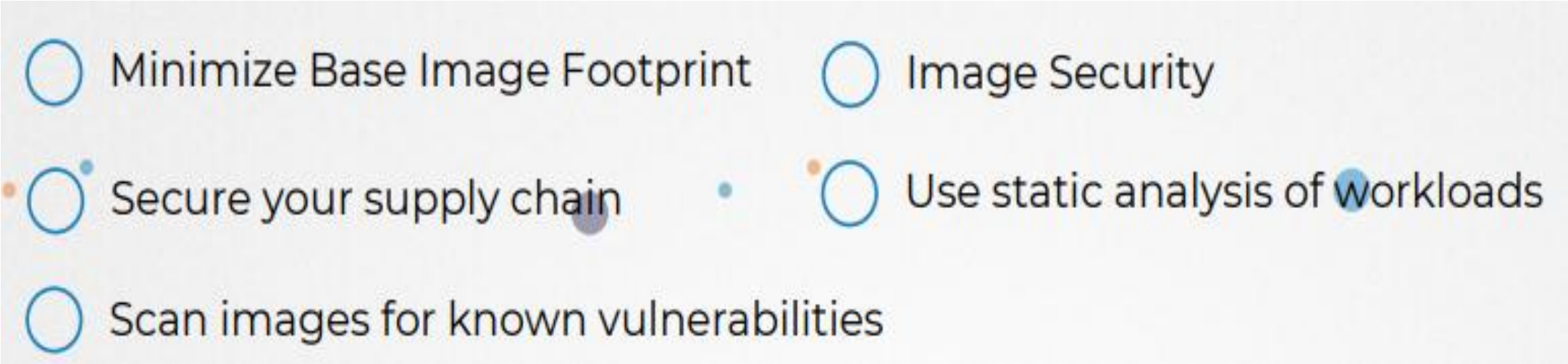


Course Objectives

- Minimize Microservice Vulnerability

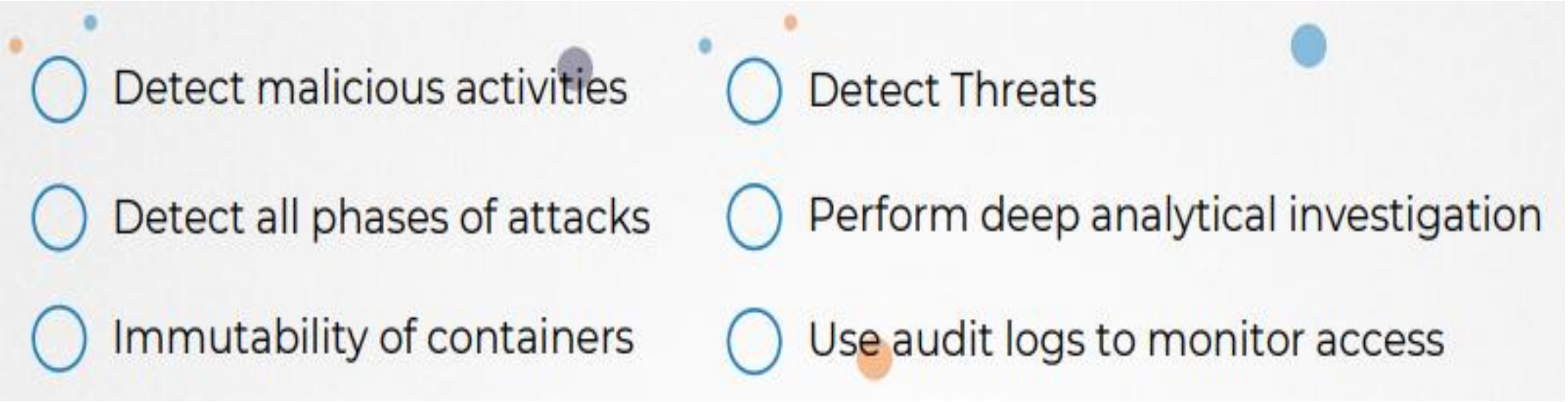
Course Objectives

- Supply Chain Security



Course Objectives

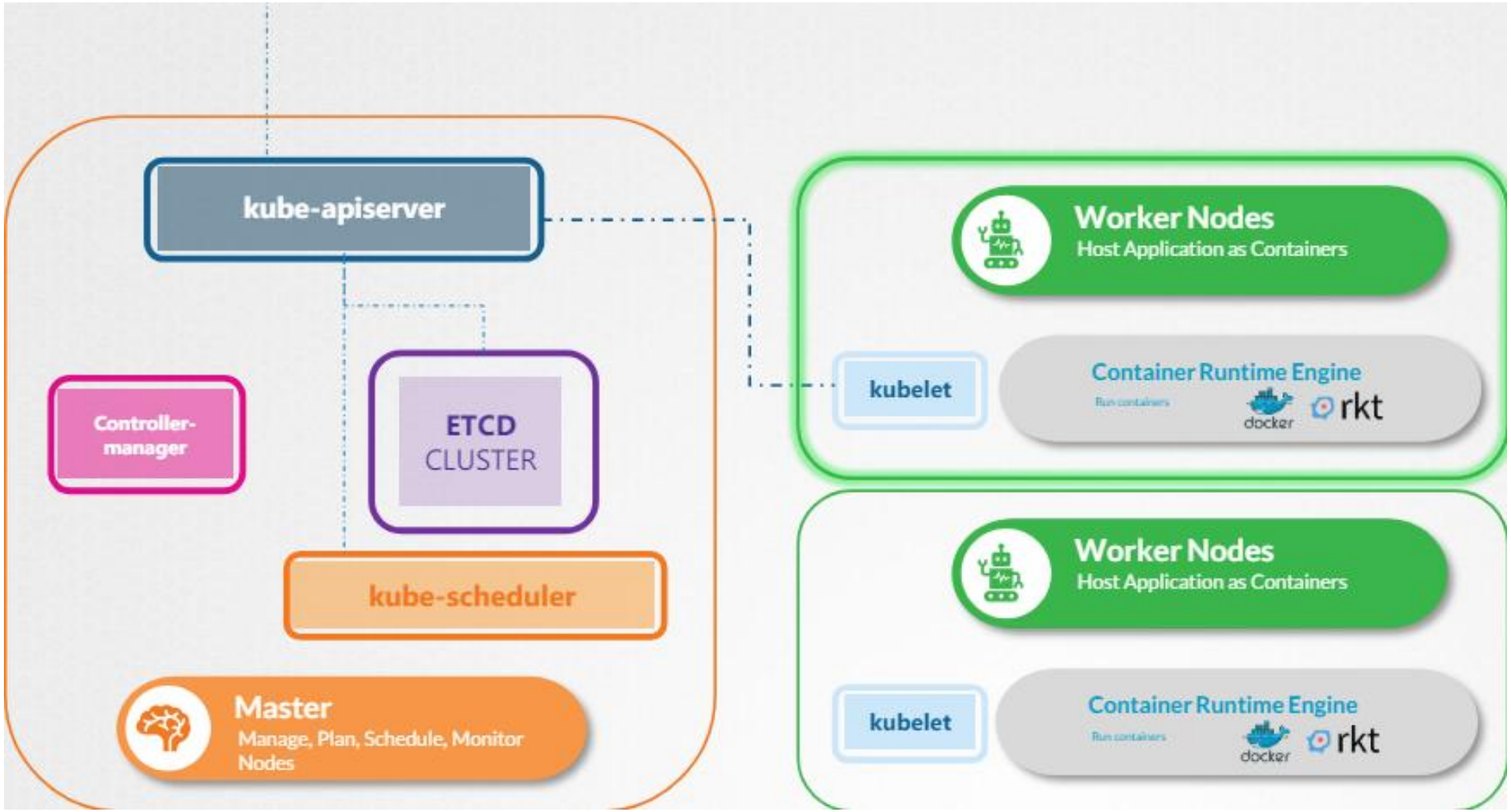
- Monitoring, Logging and Runtime Security



Exam Information

- CKA certification (non-expired) is required to sit for this exam.
- Certified Kubernetes Security Specialist (CKS): https://www.cncf.io/certification/cks/
- Exam Curriculum (Topics): https://github.com/cncf/curriculum
- Candidate Handbook: https://www.cncf.io/certification/candidate-handbook
- Exam Tips: https://docs.linuxfoundation.org/tc-docs/certification/important-instructions-cks
- Frequently asked questions for the CKS exam: https://docs.linuxfoundation.org/tc-docs/certification/faq-cka-ckad-cks

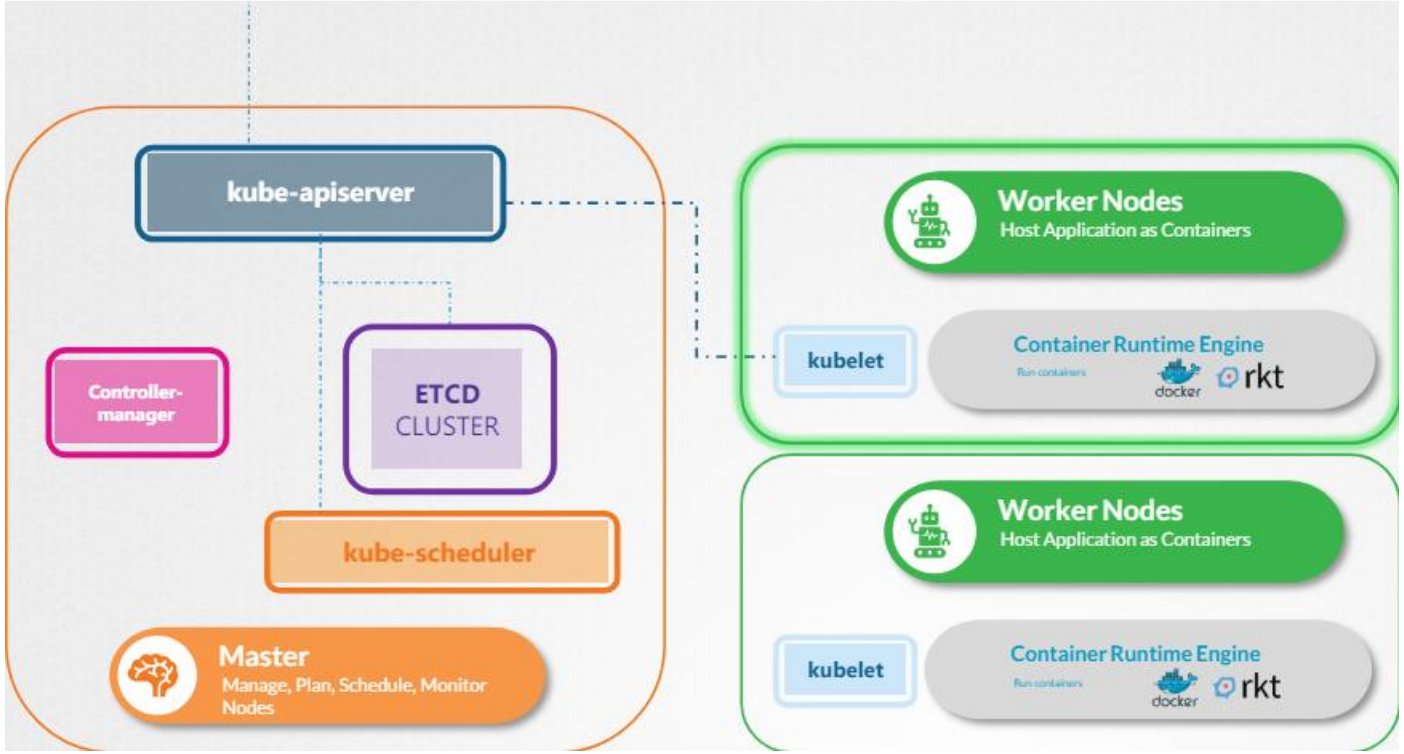
Cluster Architecture



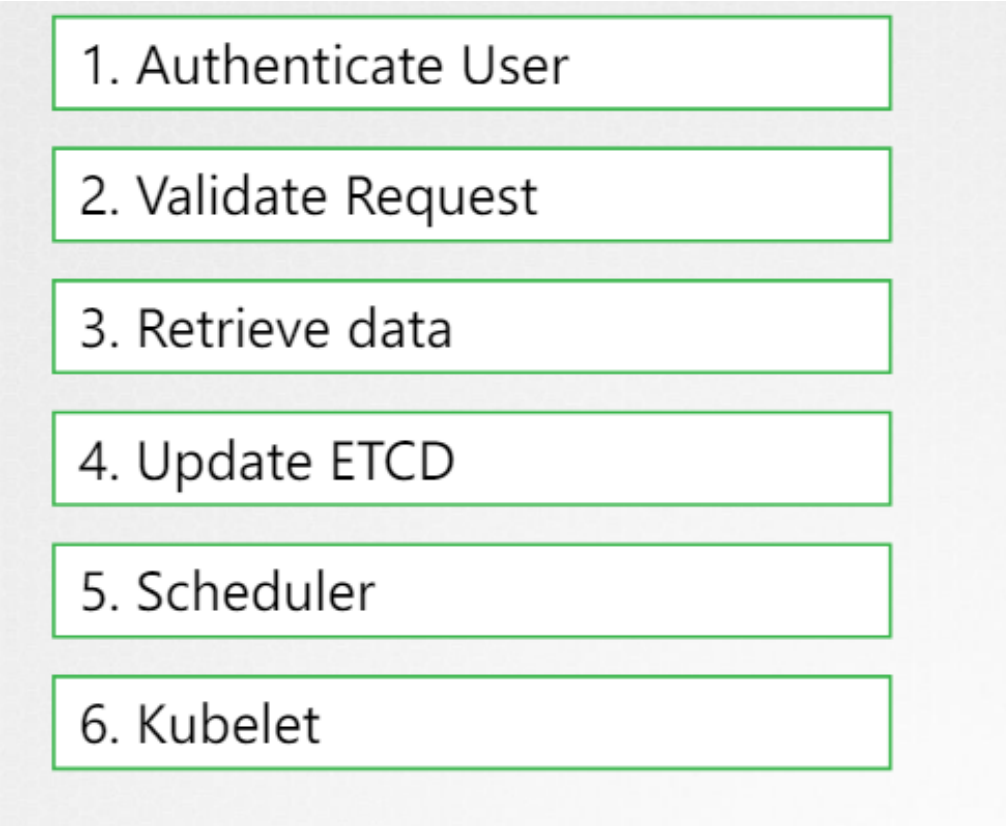
ETCD in Kubernetes

- ETCD is a distributed, reliable, key-value store that is simple, secure, and fast
- The etcd data store stores information regarding the cluster such as the nodes, pods, configs, secrets, accounts, roles, role bindings, and others
- - Version 2 commands: etcdctl backup, etcdctl cluster-health, etcdctl set, etcdctl set, etcdctl ge
- - version 3 - etcdctl snapshot save, etcdctl endpoint health, etcdctl get, etcdctl put

Kube-API server



Kube-API server



Kube-API Server

- The API server creates a pod object without assigning it to a node. Updates the information in the etcd server,updates the user that the pod has been created
- The scheduler continuously monitors the API server and realizes that there is a new pod with no node assigned. The scheduler identifies the right node to place the new pod on and communicates that back to the kube-apiserver
- The API server then updates the information in the etcd cluster
- The API server then passes that information to the kubelet in the appropriate worker node

Kube-API Server

```
cat /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
pec:
 containers:
 - command:
   - kube-apiserver
   - --authorization-mode=Node,RBAC
   - --advertise-address=172.17.0.32
   - --allow-privileged=true
   - --client-ca-file=/etc/kubernetes/pki/ca.crt
   - --disable-admission-plugins=PersistentVolumeLabel
   - --enable-admission-plugins=NodeRestriction
   - --enable-bootstrap-token-auth=true
   - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
   - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
   - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
   - --etcd-servers=https://127.0.0.1:2379
   - --insecure-port=0
   - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
   - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
   - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
   - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
   - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
   - --requestheader-allowed-names=front-proxy-client
   - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
   - --requestheader-extra-headers-prefix=X-Remote-Extra-
   - --requestheader-group-headers=X-Remote-Group
   - --requestheader-username-headers=X-Remote-User
```

Kube Controller-Manager

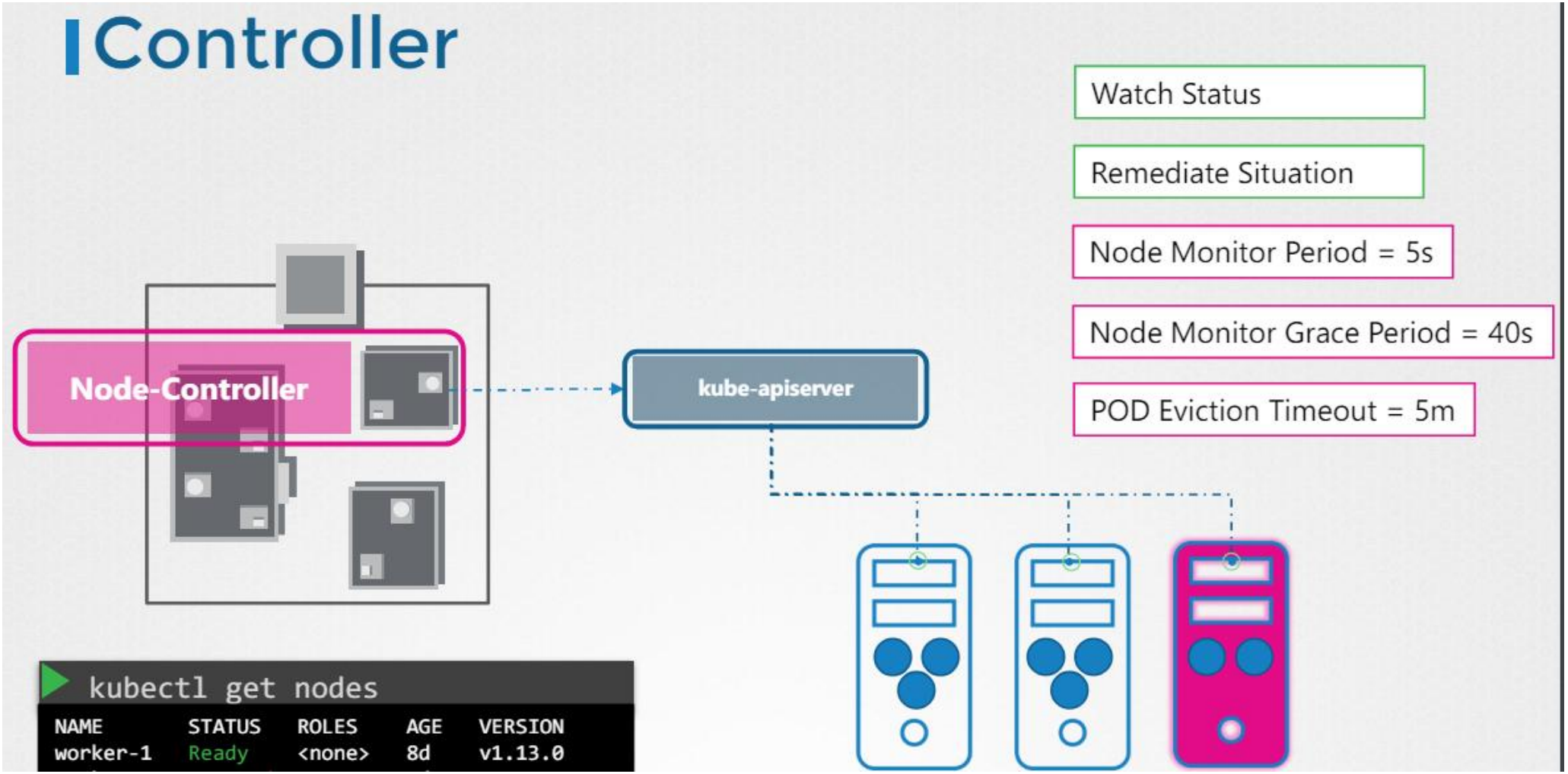
- Kube controller manager, manages various controllers in Kubernetes
- Controller is a process that continuously monitors the state of various components within the system and works towards bringing the whole system to the desired functioning state
- For example, the node controller is responsible for monitoring the status of the nodes and taking necessary actions to keep the applications running

Kube Controller-Manager

```
cat /etc/kubernetes/manifests/kube-controller-manager.yaml
```

```
spec:
  containers:
  - command:
    - kube-controller-manager
    - --address=127.0.0.1
    - --cluster-signing-cert-file=/etc/kubernetes/pki/ca.crt
    - --cluster-signing-key-file=/etc/kubernetes/pki/ca.key
    - --controllers=*,bootstrapsigner,tokencleaner
    - --kubeconfig=/etc/kubernetes/controller-manager.conf
    - --leader-elect=true
    - --root-ca-file=/etc/kubernetes/pki/ca.crt
    - --service-account-private-key-file=/etc/kubernetes/pki/sa.key
    - --use-service-account-credentials=true
```

Node Controller



Kube Scheduler

- The Kubernetes scheduler is responsible for scheduling pods on nodes
- scheduler is only responsible for deciding which pod goes on which node.It doesn't actually place the pod on the nodes. That's the job of the kubelet
- You may have pods with different resource requirements.It may have a set of CPU and memory requirements
- The scheduler ranks the nodes to identify the best fit for the pod. It uses a priority function to assign a score to the nodes on a scale of zero to 10

Kube Scheduler

```
cat /etc/kubernetes/manifests/kube-scheduler.yaml
```

```
spec:
  containers:
  - command:
    - kube-scheduler
    - --address=127.0.0.1
    - --kubeconfig=/etc/kubernetes/scheduler.conf
    - --leader-elect=true
```

Kubelet

- The kubelet in the Kubernetes worker node registers the node with a Kubernetes cluster
- The kubelet continues to monitor the state of the pod and containers in it and reports to the kube API server on a timely basis
- If you use the kubeadm tool to deploy your cluster,it does not automatically deploy the kubelet

Kubelet

kubelet.service

```
ExecStart=/usr/local/bin/kubelet \\
  --config=/var/lib/kubelet/kubelet-config.yaml \\
  --container-runtime=remote \\
  --container-runtime-endpoint=unix:///var/run/containerd/containerd.sock \\
  --image-pull-progress-deadline=2m \\
  --kubeconfig=/var/lib/kubelet/kubeconfig \\
  --network-plugin=cni \\
  --register-node=true \\
  --v=2
```

Kube proxy

- Kube-proxy is a process that runs on each node in the Kubernetes cluster
- Its job is to look for new services, and every time a new service is created, it creates the appropriate rules on each node to forward traffic on those services to the pods
- Kube-proxy is deployed as a DaemonSet, so a single pod is always deployed on each node in the cluster

PODs

- Kubernetes does not deploy application containers directlyon the worker nodes.
- The containers are encapsulated into a Kubernetes object known as pods
- pods usually have a one-to-one relationship with the containers but a single pod can have multiple containers
- Deploy a pod: kubectl run nginx --image nginx

Creating PODs declaratively

- Kubernetes uses YAML files as inputs for the creation of objects such as pods, replicas, deployments, services etc
- 4 top level fields in every yaml are the API version, kind, metadata, and spec
- apiVersion: This is the version of the Kubernetes API
- kind: type of the object (pod,replica-set,deployment,service etc)
- metadata: data about the object like its name, labels etc. Its a disctionary
- spec: The specification section which is written as spec. Depending on the object we are going to create, this is where we would provide additional information to Kubernetes pertaining to that object

Replica sets

- Controllers are the brain behind Kubernetes. They are the processes that monitor Kubernetes objects and respond accordingly
- The Replication Controller helps us run multiple instances of a single pod in the Kubernetes cluster,thus providing high availability
- Replication Controller is the older technology that is being replaced by Replica Set
- version in replication controller is v1 vs apps/v1 in replica set

Replica sets

- version in replication controller is v1 vs apps/v1 in replica set
- kind in replication controller is ReplicationController vs ReplicaSet in replica set
- template is same for both but selector is optional in replication controller but mandatory in replica set
- The selector section helps the Replica Set identify what pods fall under it
- Upgrade replica set kubectl scale --replicas=6 replicaset myapp-replicaset

Deployment

- Deployment is a Kubernetes object that comes higher in the hierarchy of replica set and pods
- The deployment provides us with the capability to upgrade the underlying instances seamlessly using rolling updates, undo changes, and pause, and resume changes as required
- Imperative creation: kubectl create deployment --image=httpd:1.4-alpine httpd-frontend --replicas=3
- To edit the deployment: kubectl edit deployment web-app

Services: Nodeport

- Kubernetes Services enable communication between various components within and outside of the application
- NodePort service is the service that listens to a port on the node and forward request to the Pods
- The service is like a virtual server inside the node. Inside the cluster it has its own IP address called ClusterIP of service
- target port is the port on the Pod where the actual container is running
- The second port is the port on the service itself
- Finally, node port is the port on the node itself
- When pods are distributed across multiple nodes k8s creates a service which spans across all the nodes

Service: ClusterIP

- We cannot rely on these IP addresses for internal communication between the applications because IP address assigned to PODs can change
- A Kubernetes ClusterIP service helps us group the pods together and provide a single interface to access the pods in a group
- This enables us to easily and effectively deploy a microservices based application on Kubernetes cluster

Services: Nodeport

- Kubernetes Services enable communication between various components within and outside of the application
- NodePort service is the service that listens to a port on the node and forward request to the Pods
- The service is like a virtual server inside the node. Inside the cluster it has its own IP address called ClusterIP of service
- target port is the port on the Pod where the actual container is running
- The second port is the port on the service itself
- Finally, node port is the port on the node itself
- When pods are distributed across multiple nodes k8s creates a service which spans across all the nodes

Service: LoadBalancer

- To load balance requests on the all nodes in the cluster where pods are running
- Only usable on cloud platforms with support of native load balancer (eg AWS, GCP, Azure etc)
- Service definition is similar to NodePort except type: LoadBalancer

Namespace

- Whatever Kubernetes objects we created such as pods, deployments, and services in our cluster. we have been doing all this within a name space. This name space is known as the default name space
- Kubernetes creates a set of pods and services for its internal purpose
- To isolate these from the user and to prevent you from accidentally deleting or modifying these services, Kubernetes creates them under another name space created at cluster startup named kube-system
- One example of using a namespace is if you wanted to use the same cluster for both dev and production environment,but at the same time, isolate the resources between them

Namespace

- You can also assign quota of resources to each of these name spaces. That way, each name space is guaranteed a certain amount and does not use more than its allowed limit
- the resources within a name space can refer to each other simply by their names
- kubectl get pods --namespace=kube-system
- To switch to a different namespace: kubectl config set-context $(kubectl config current-context) --namespace=dev
- To get pods from all namespace: kubectl get pods --all-namespaces

Cluster Maintenance: OS Upgrade

- you might have to take down nodes as part of your cluster, say for maintenance purposes, like upgrading a software
- If the node was down for more than five minutes, then the pods are terminated from that node
- The time it waits for a pod to come back online is known as the pod-eviction-timeout and is set on the controller manager with a default value of five minutes
- command for drain is kubectl drain node-1
- kubectl uncordon node-1
- you can manually mark a node cordoned using kubectl cordon node-1

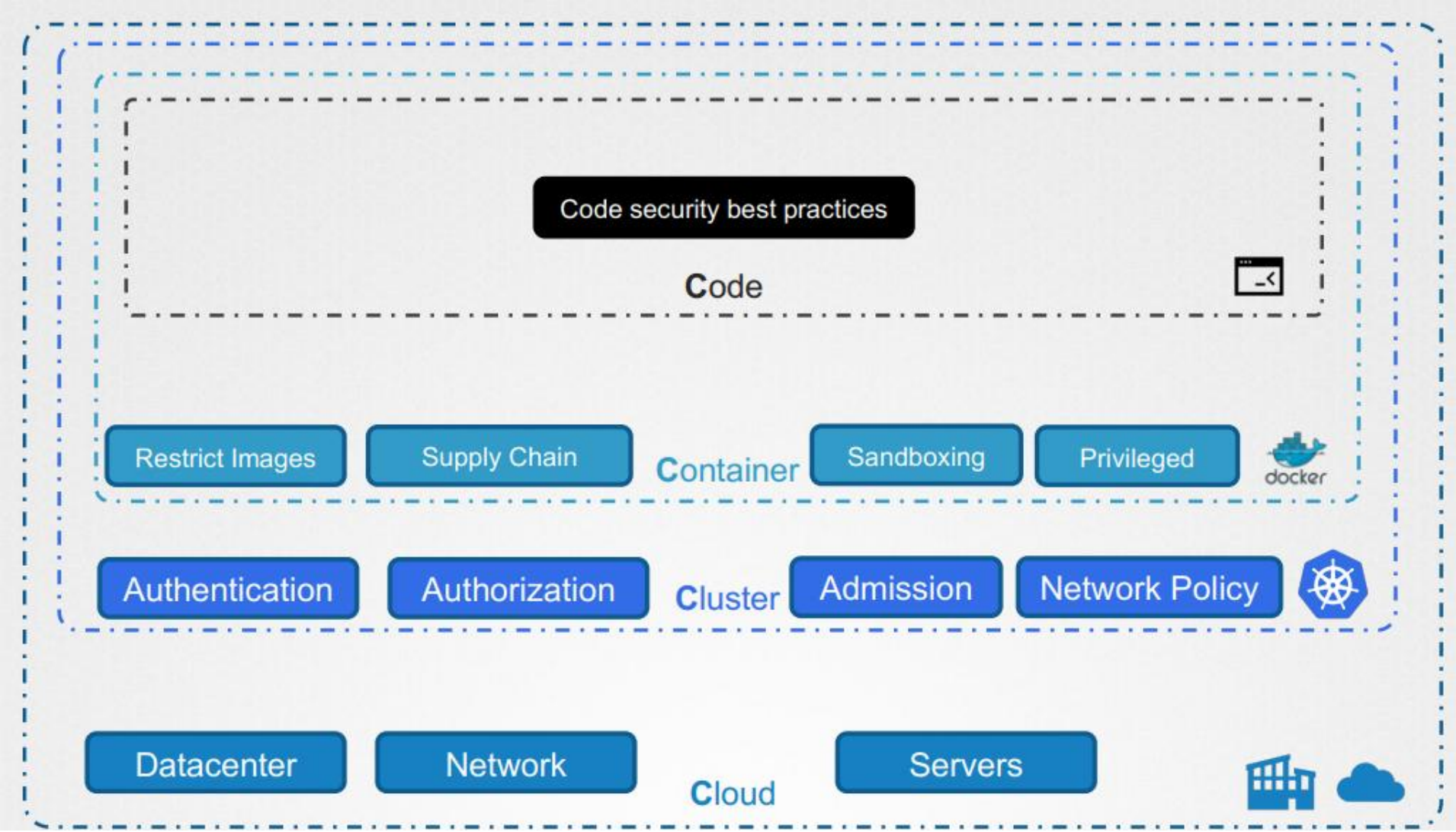
Cluster Upgrade process

- Since the kube-apiserver is the primary component in the control plane, and that is the component that all other components talk to,none of the other components should ever be at a version higher than the kube-apiserver
- The controller-manager and scheduler can be at one version lower
- kubelet and kube-proxy components can be at two versions lower
- kubectl utility could 1 version higher, same or 1 version lower than api-server
- Kubernetes supports only up to the recent three minor versions, if current is 1.29 then 1.28,1.27 are supported

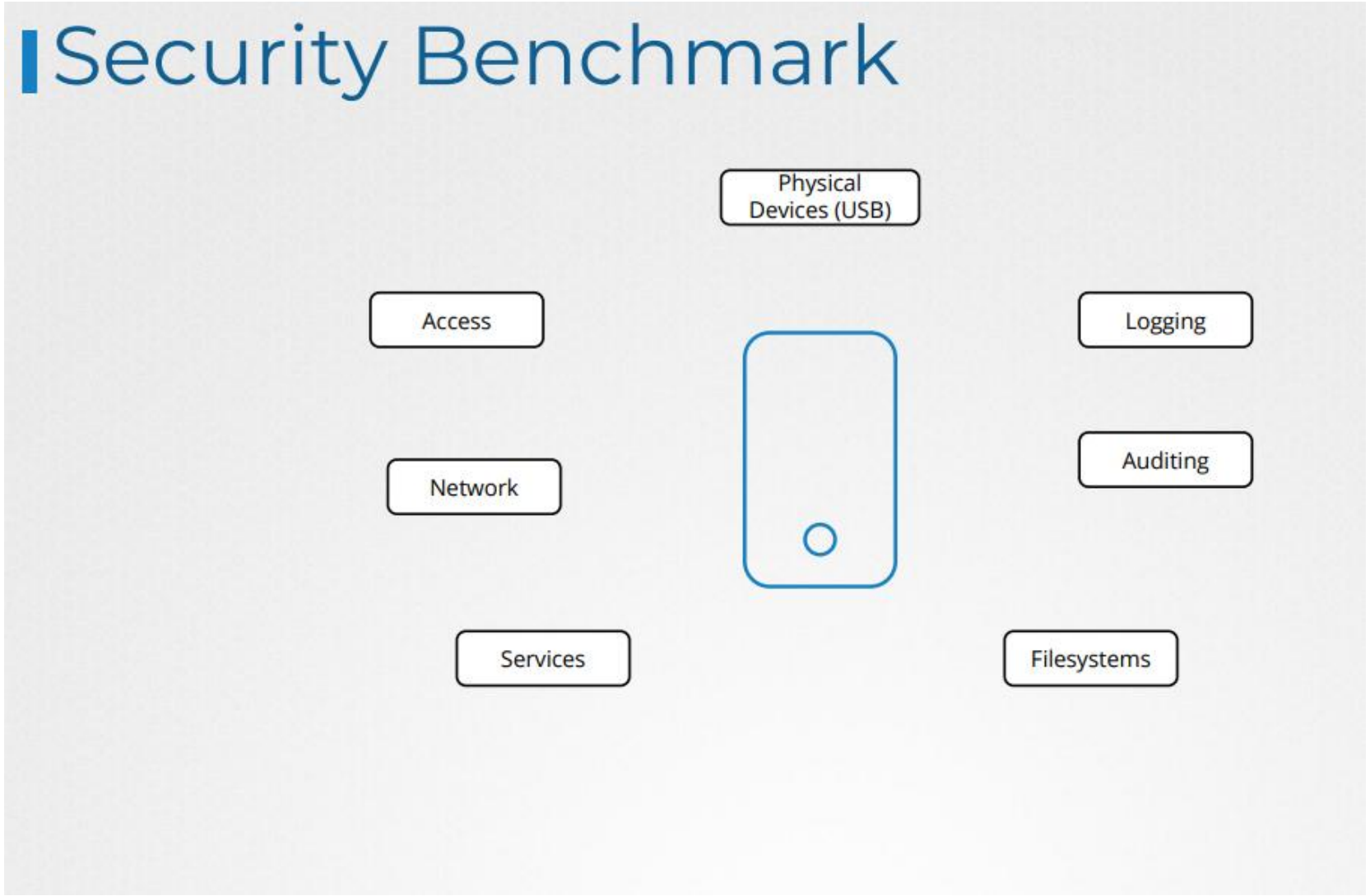
Cluster Upgrade process

- tools like kubeadm, then the tool can help you plan and upgrade the cluster
- First, you upgrade your master nodes and then upgrade the worker nodes
- kubeadm has an upgrade command, it will give you a lot of good information, you must upgrade the kubeadm tool itself before you can upgrade the cluster

The 4 C's of Cloud Native Security



CIS Benchmark



CIS Benchmark

CIS Benchmark

OS	Cloud	Mobile	Network	Desktop	Server
Linux	Google	Apple IOS	Check Point	Web Browsers	Tomcat
Windows	Azure	Android	Cisco	MS Office	Docker
MacOS	AWS		Juniper	Zoom	Kubernetes
More..	More..		Palo Alto Networks	More..	More..

https://www.cisecurity.org/cis-benchmarks/

CIS Benchmark for kubernetes

CIS Center for Internet Security®

kube-bench

```
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 API Server
[FAIL] 1.1.1 Ensure that the --allow-privileged argument is se
[FAIL] 1.1.2 Ensure that the --anonymous-auth argument is set
[PASS] 1.1.3 Ensure that the --basic-auth-file argument is not
[PASS] 1.1.4 Ensure that the --insecure-allow-any-token argume
[FAIL] 1.1.5 Ensure that the --kubelet-https argument is set t
[PASS] 1.1.6 Ensure that the --insecure-bind-address argument
[PASS] 1.1.7 Ensure that the --insecure-port argument is set t
[PASS] 1.1.8 Ensure that the --secure-port argument is not set
[FAIL] 1.1.9 Ensure that the --profiling argument is set to fa
[FAIL] 1.1.10 Ensure that the --repair-malformed-updates argum
[PASS] 1.1.11 Ensure that the admission control policy is not
[FAIL] 1.1.12 Ensure that the admission control policy is set
[FAIL] 1.1.13 Ensure that the admission control policy is set
[FAIL] 1.1.14 Ensure that the admission control policy is set
[PASS] 1.1.15 Ensure that the admission control policy is set
[FAIL] 1.1.16 Ensure that the --audit-log-path argument is set
[FAIL] 1.1.17 Ensure that the --audit-log-maxage argument is s
[FAIL] 1.1.18 Ensure that the --audit-log-maxbackup argument i
[FAIL] 1.1.19 Ensure that the --audit-log-maxsize argument is
[PASS] 1.1.20 Ensure that the --authorization-mode argument is
[PASS] 1.1.21 Ensure that the --token-auth-file parameter is n
[FAIL] 1.1.22 Ensure that the --kubelet-certificate-authority
```

CIS Benchmark Kube-bench

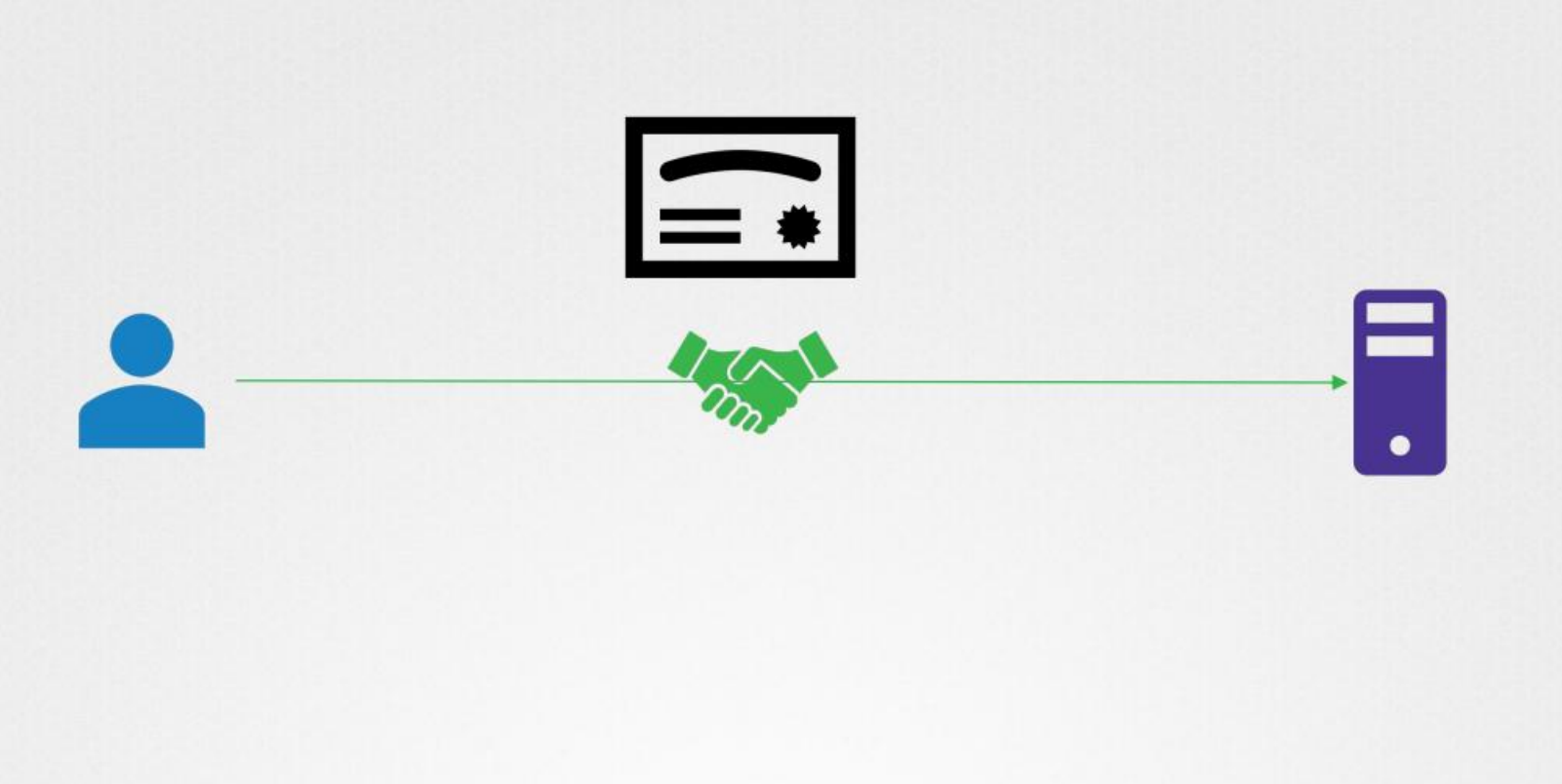
Deploy Kube-bench

- Deploy as a Docker Container
- Deploy as a POD in a Kubernetes cluster
- Install kube-bench binaries
- Compile from source

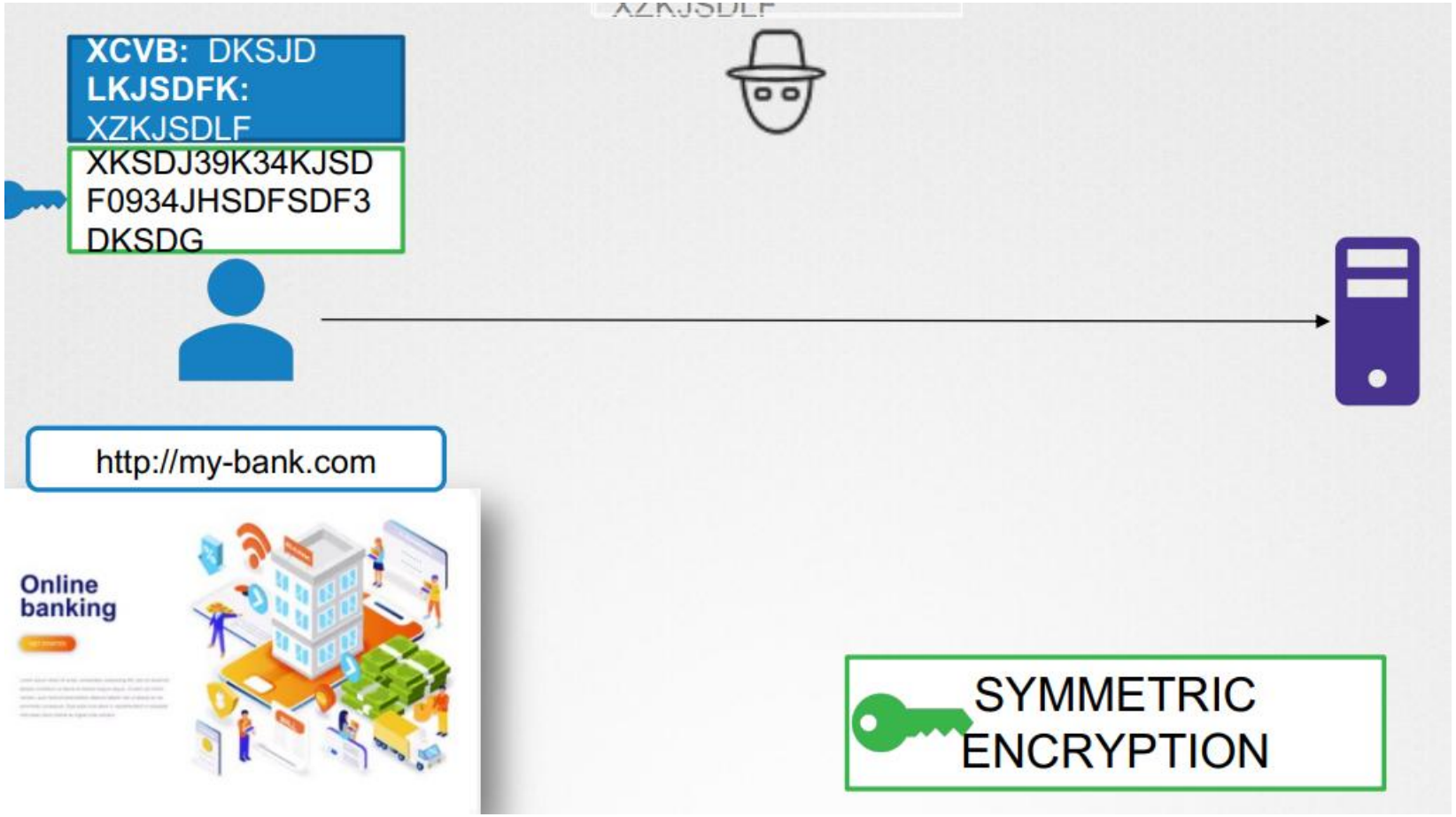
CIS Benchmark Kube-bench

- Go to kube-bench github page
- Identify latest release binary
- Install kube-bench on master node
- Run assessment and review results
- Fix issues

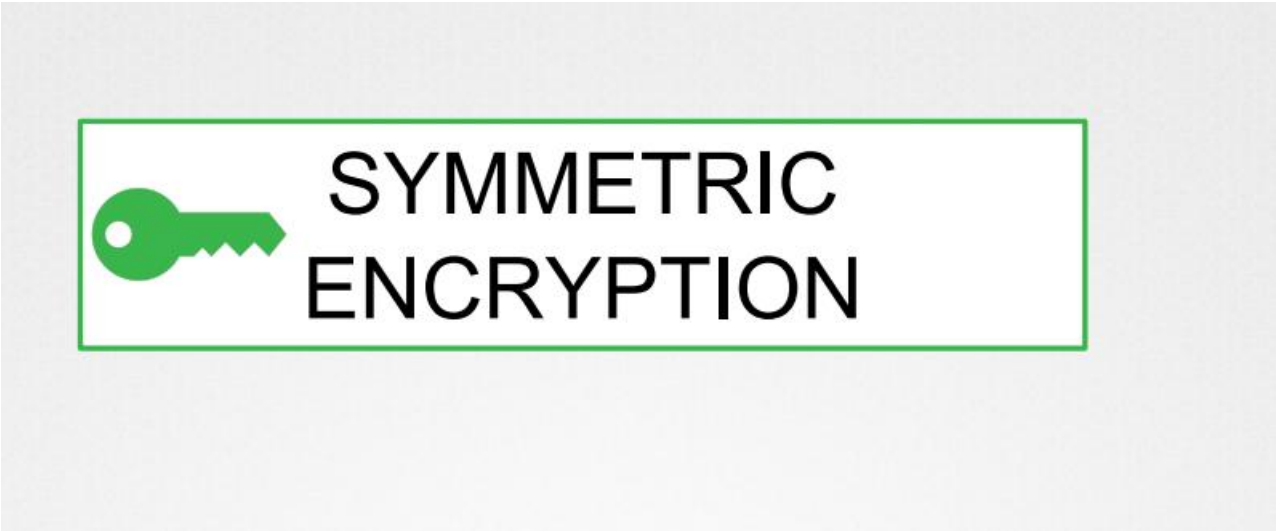
TLS Basics



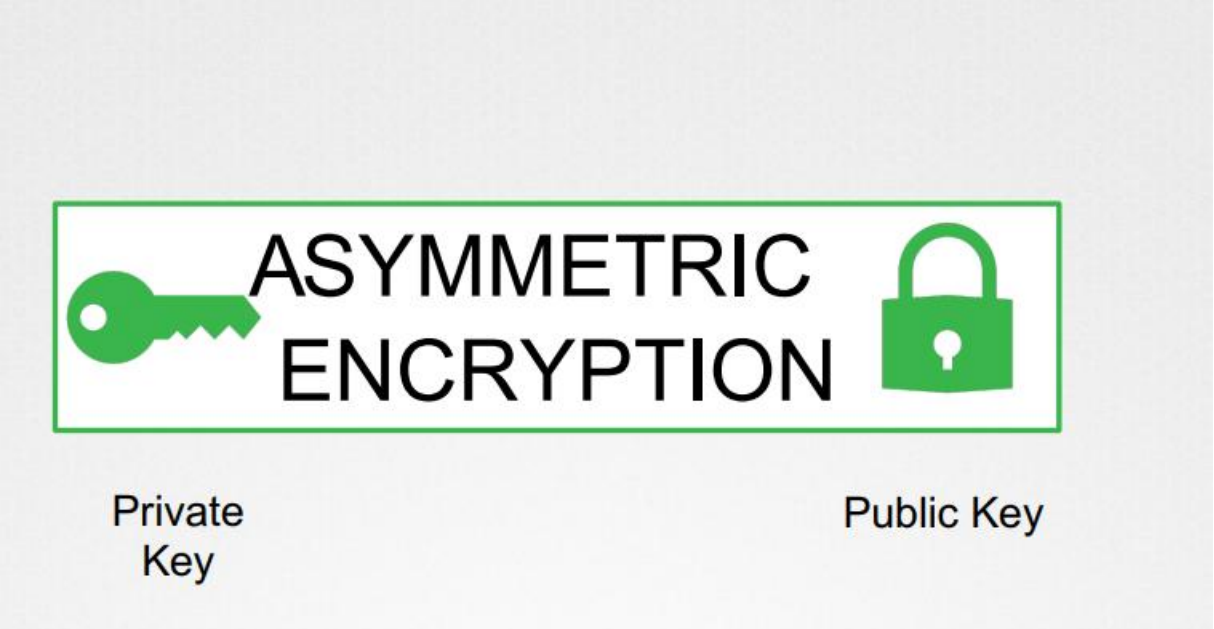
TLS Basics


XCVB: DKSJD
LKJSDFK:
XZKJSDLF
XKSDJ39K34KJSD
F0934JHSDFSDF3
DKSDG
http://my-bank.com
Online
banking
SYMMETRIC
ENCRYPTION

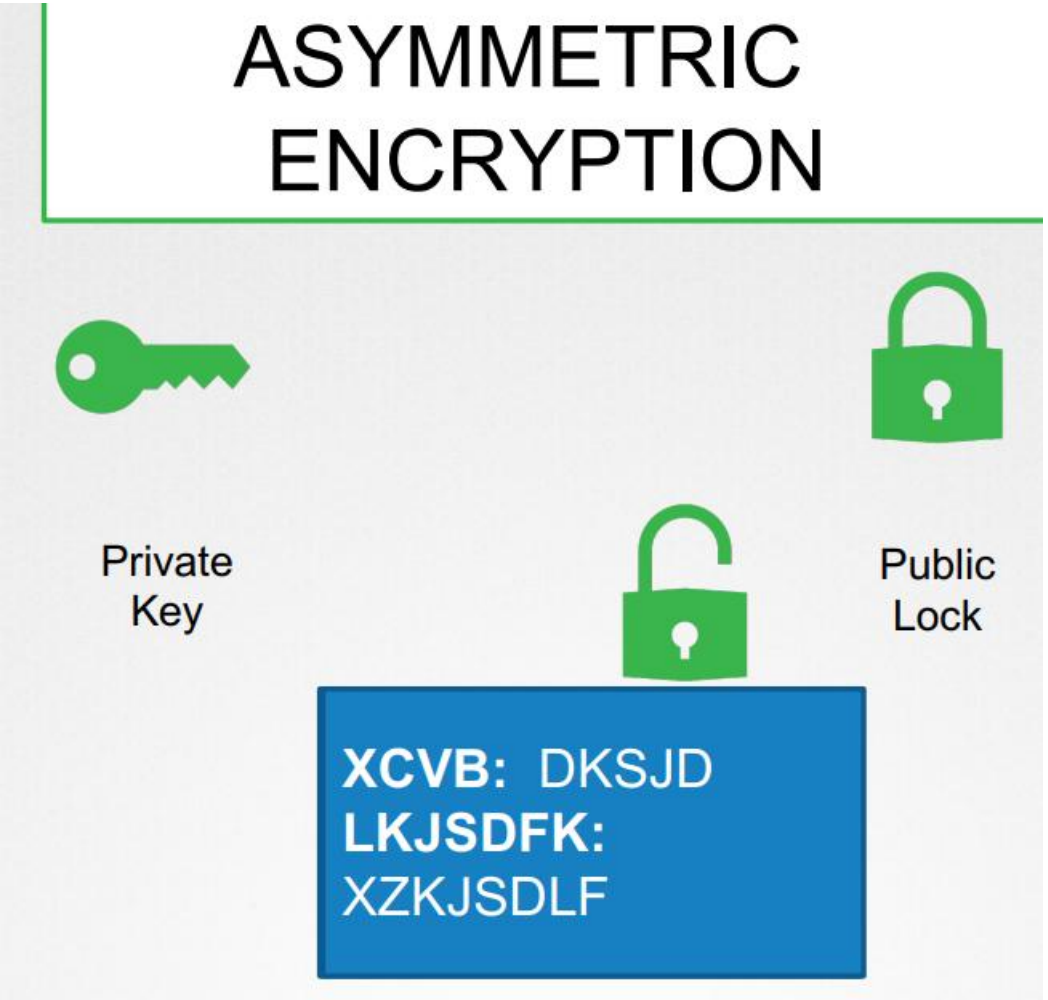
TLS Basics



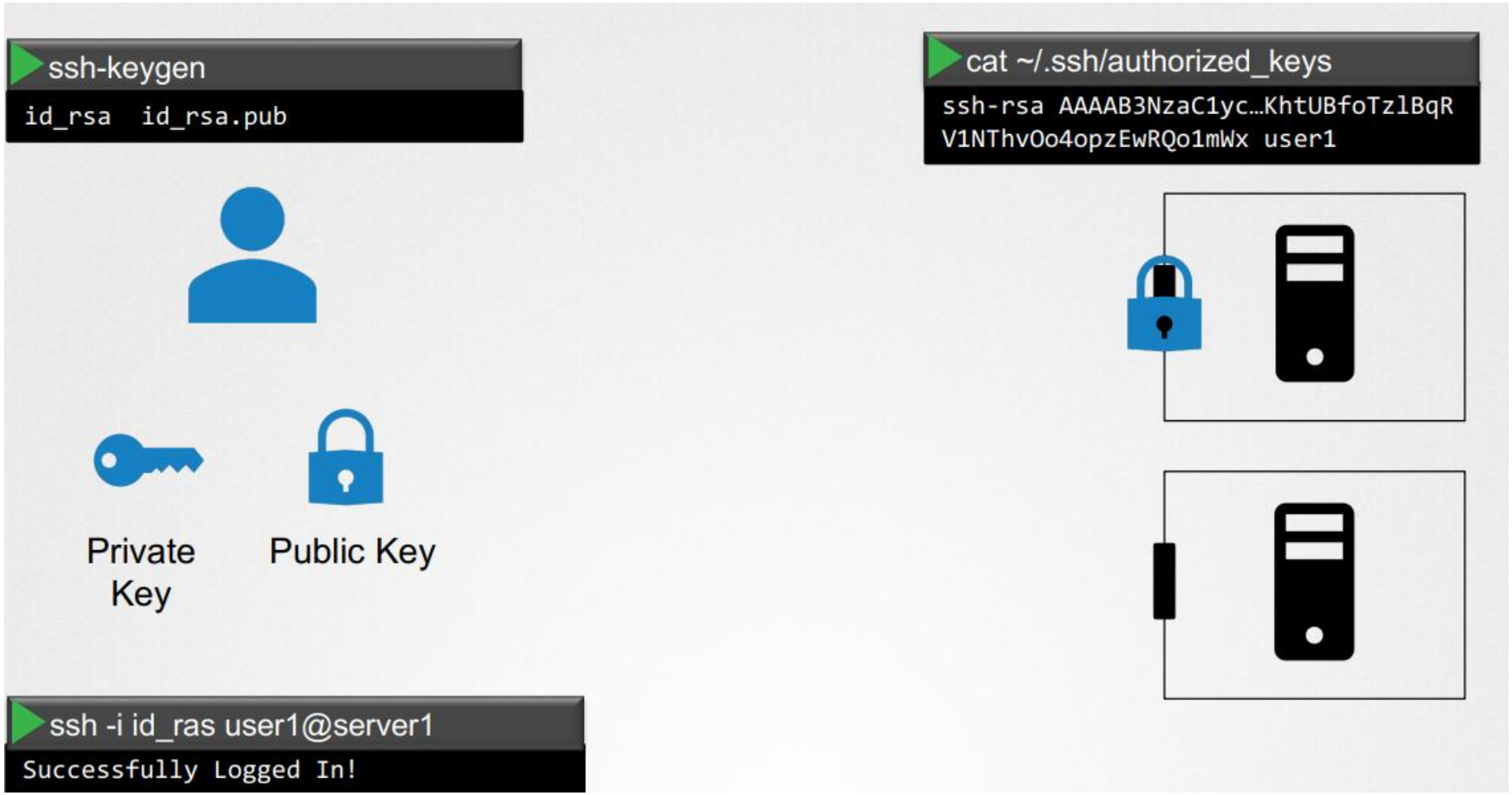
TLS Basics



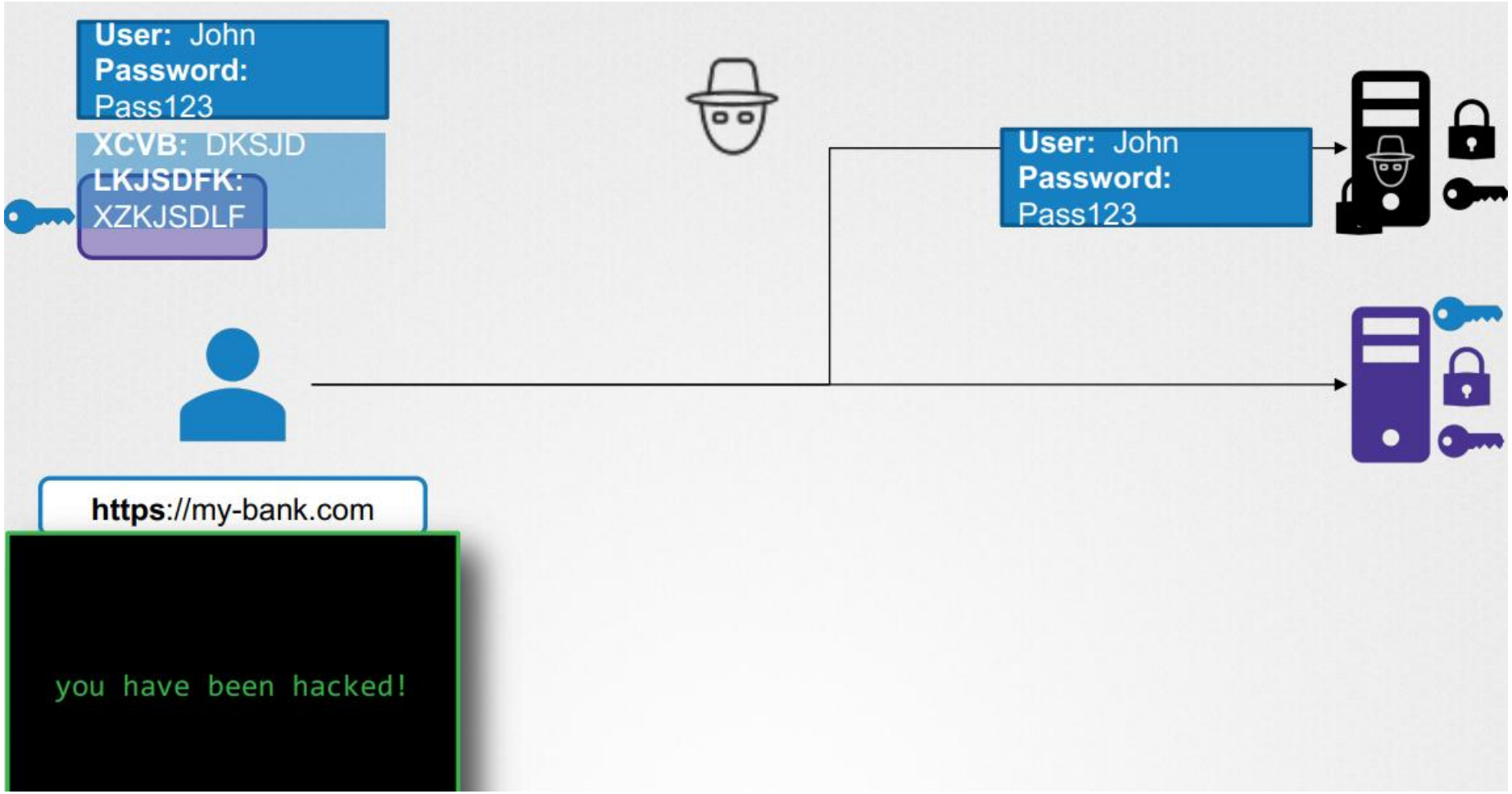
TLS Basics


ASYMMETRIC
ENCRYPTION
Private
Key
Public
Lock
XCVB: DKSJD
LKJSDFK:
XZKJSDLF

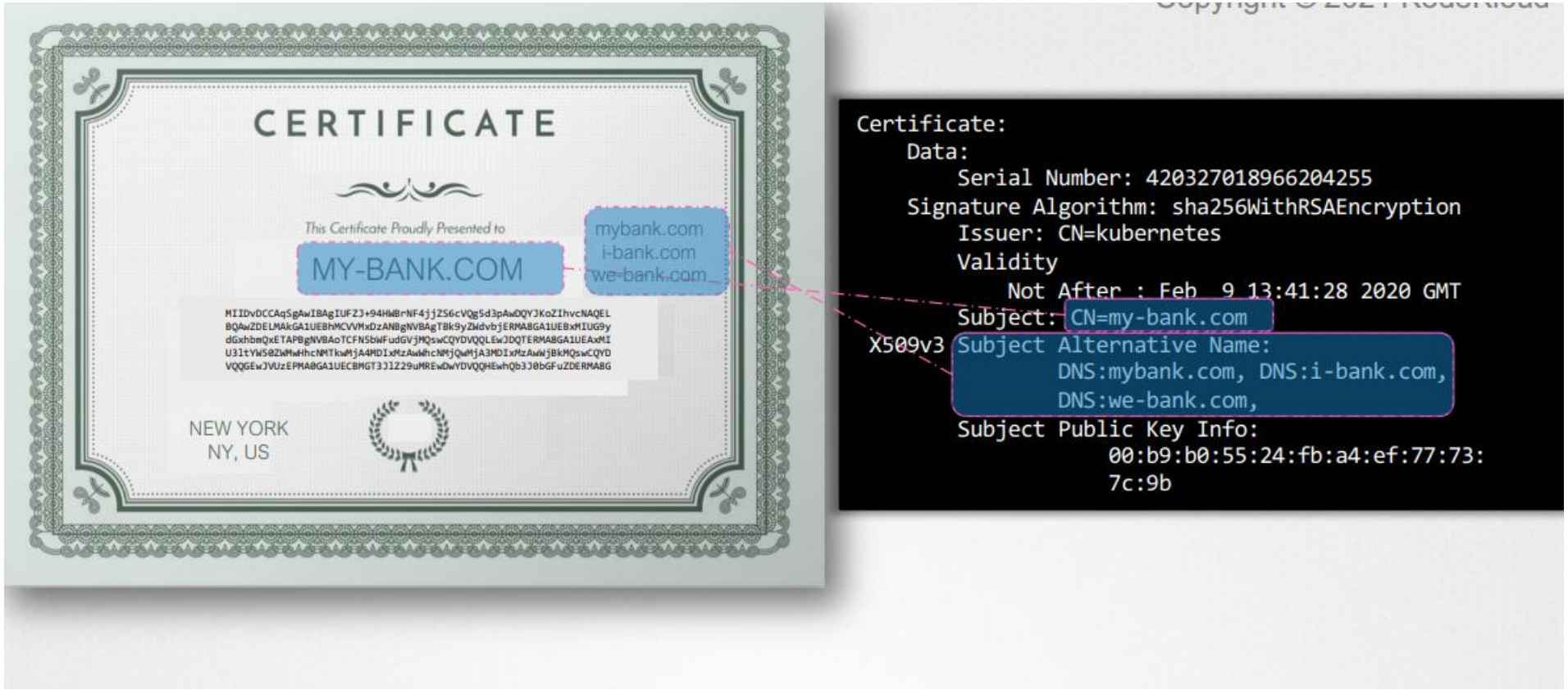
TLS Basics


ssh-keygen
id_rsa id_rsa.pub
Private Key
Public Key
cat ~/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc…KhtUBfoTzlBqR
V1NThvOo4opzEwRQo1mWx user1
ssh -i id_ras user1@server1
Successfully Logged In!

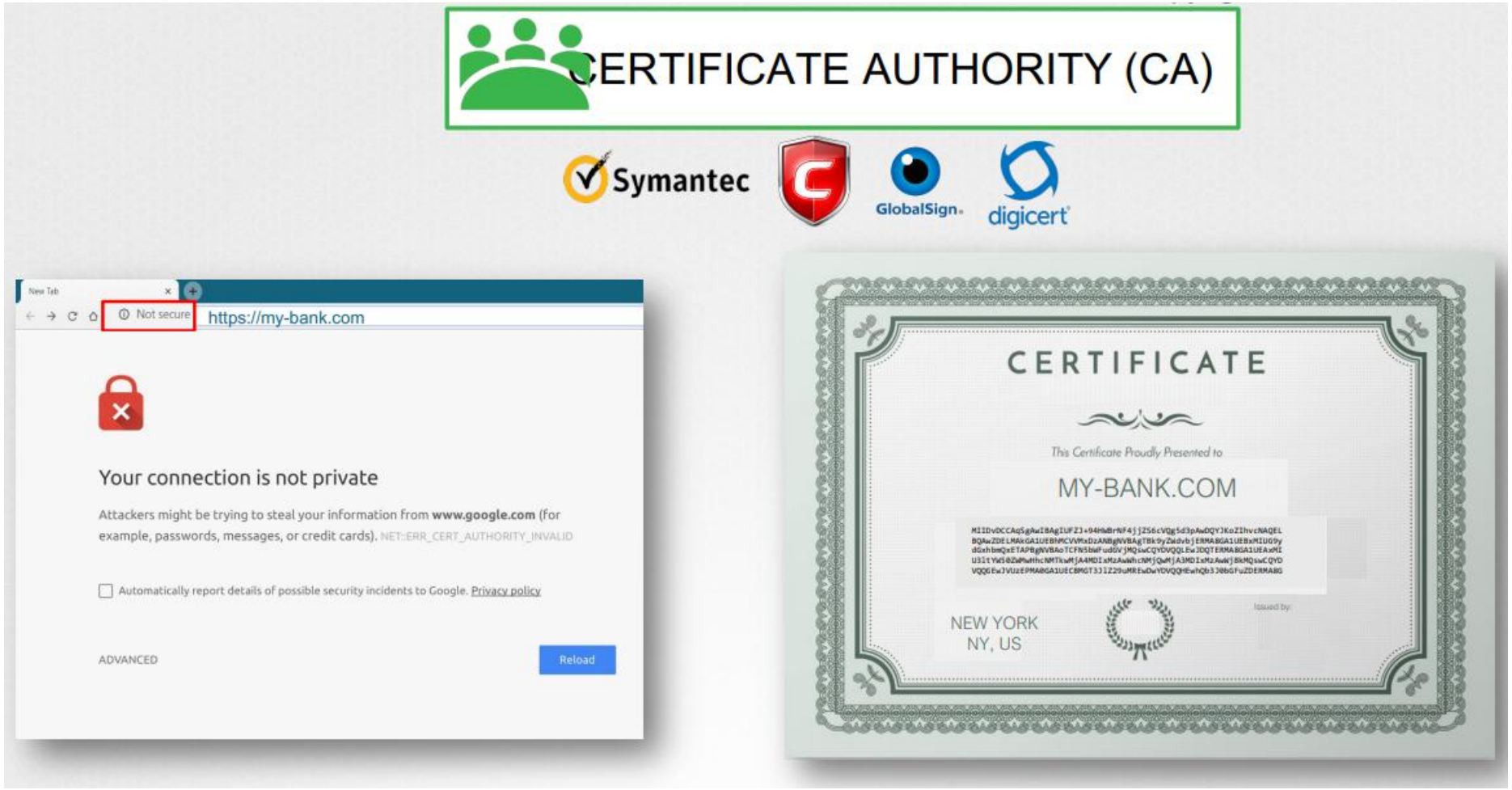
TLS Basics


User: John
Password:
Pass123
XCVB: DKSJD
LKJSDFK:
XZKJSDLF
User: John
Password:
Pass123
https://my-bank.com
you have been hacked!

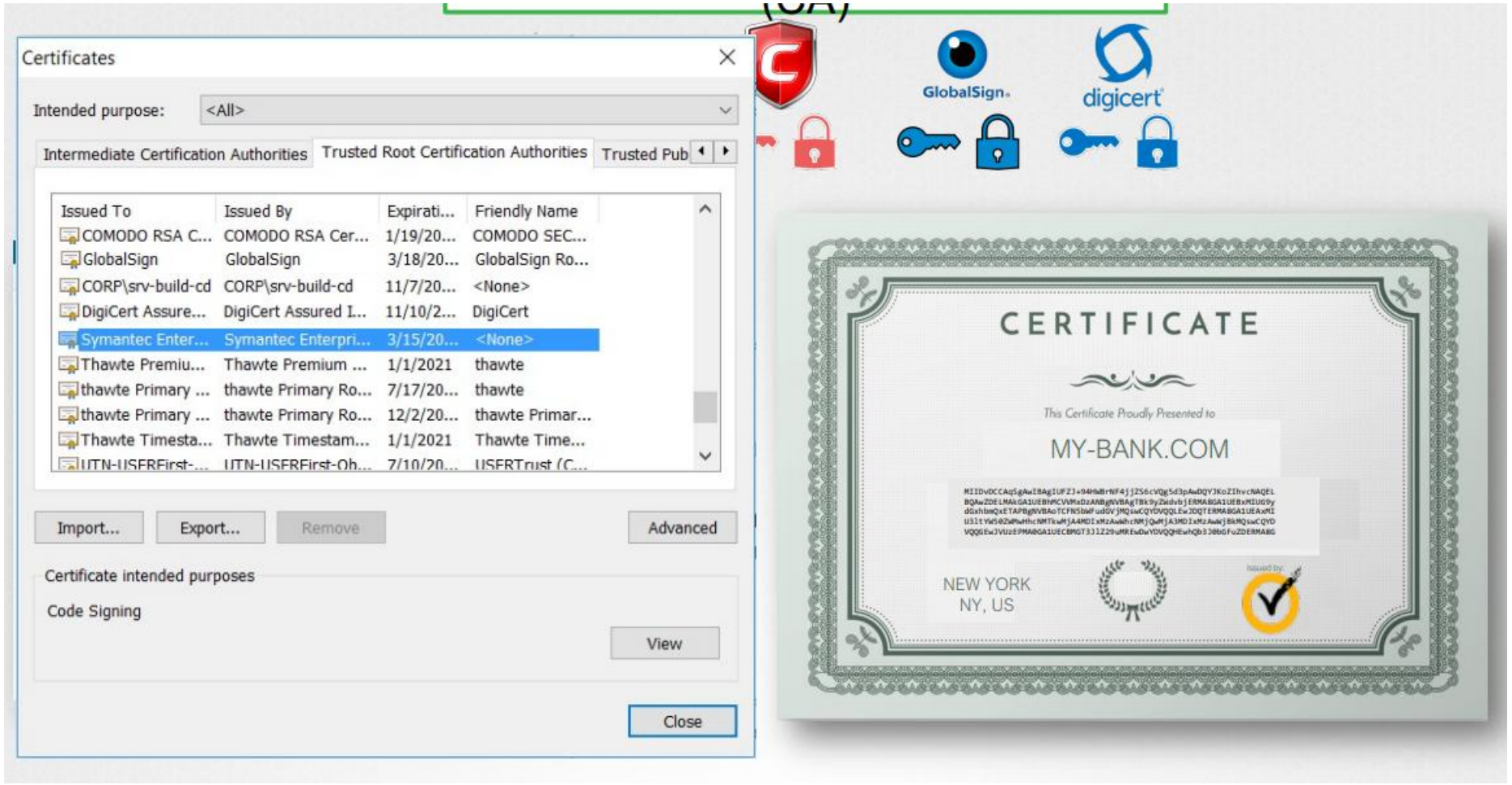
TLS Basics


CERTIFICATE
This Certificate Proudly Presented to
MY-BANK.COM
mybank.com
i-bank.com
we-bank.com
MIIDvDCCAqSgAwIBAgIUFZJ+94HWBrNF4jjZS6cVQg5d3pAwDQYJKoZIhvcNAQEL
BQAwZDELMAkGA1UEBhMCVVMxDzANBgNVBAgTBk9yZWdvbjERMA8GA1UEBxMIUG9y
dGxhbmQxETAPBgNVBAoTCFN5bWFudGVjMQswCQYDVQQLEwJDQTERMA8GA1UEAxMI
U3ltYWS0ZWMwHhcNMTkwMjA4MDIxMzAwWhcNMjQwMjA3MDIxMzAwWjBkMQswCQYD
VQQGEwJVUzEPMA0GA1UECBMGT3JlZ29uMREwDwYDVQQHEwhQb3J0bGFuZDERMA8G
NEW YORK
NY, US
Certificate:
Data:
Serial Number: 420327018966204255
Signature Algorithm: sha256WithRSAEncryption
Issuer: CN=kubernetes
Validity
Not After : Feb 9 13:41:28 2020 GMT
Subject: CN=my-bank.com
X509v3 Subject Alternative Name:
DNS:mybank.com, DNS:i-bank.com,
DNS:we-bank.com,
Subject Public Key Info:
00:b9:b0:55:24:fb:a4:ef:77:73:
7c:9b

TLS Basics


CERTIFICATE AUTHORITY (CA)
Symantec
GlobalSign
digicert
New Tab
Not secure
https://my-bank.com
Your connection is not private
Attackers might be trying to steal your information from www.google.com (for example, passwords, messages, or credit cards). NET::ERR_CERT_AUTHORITY_INVALID
Automatically report details of possible security incidents to Google. Privacy policy
ADVANCED
Reload
CERTIFICATE
This Certificate Proudly Presented to
MY-BANK.COM
NEW YORK
NY, US

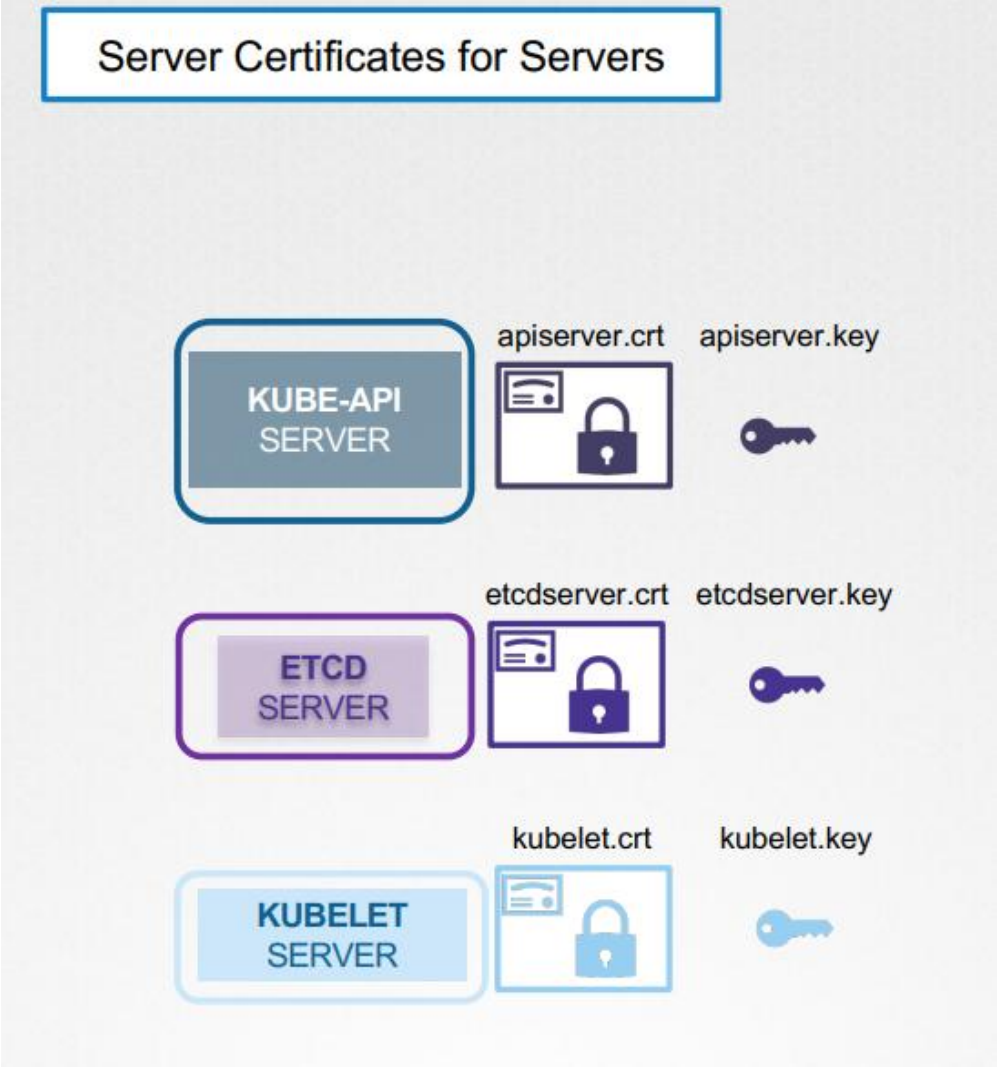
TLS Basics


Certificates
Intended purpose:
<All>
Intermediate Certification Authorities
Trusted Root Certification Authorities
Trusted Pub
Issued To
Issued By
Expirati...
Friendly Name
COMODO RSA C...
COMODO RSA Cer...
1/19/20...
COMODO SEC...
GlobalSign
GlobalSign
3/18/20...
GlobalSign Ro...
CORP\srv-build-cd
CORP\srv-build-cd
11/7/20...
<None>
DigiCert Assure...
DigiCert Assured I...
11/10/2...
DigiCert
Symantec Enter...
Symantec Enterpri...
3/15/20...
<None>
Thawte Premiu...
Thawte Premium ...
1/1/2021
thawte
thawte Primary ...
thawte Primary Ro...
7/17/20...
thawte
thawte Primary ...
thawte Primary Ro...
12/2/20...
thawte Primar...
Thawte Timesta...
Thawte Timestam...
1/1/2021
Thawte Time...
Import...
Export...
Remove
Advanced
Certificate intended purposes
Code Signing
View
Close
GlobalSign
digicert
CERTIFICATE
This Certificate Proudly Presented to
MY-BANK.COM
NEW YORK
NY, US

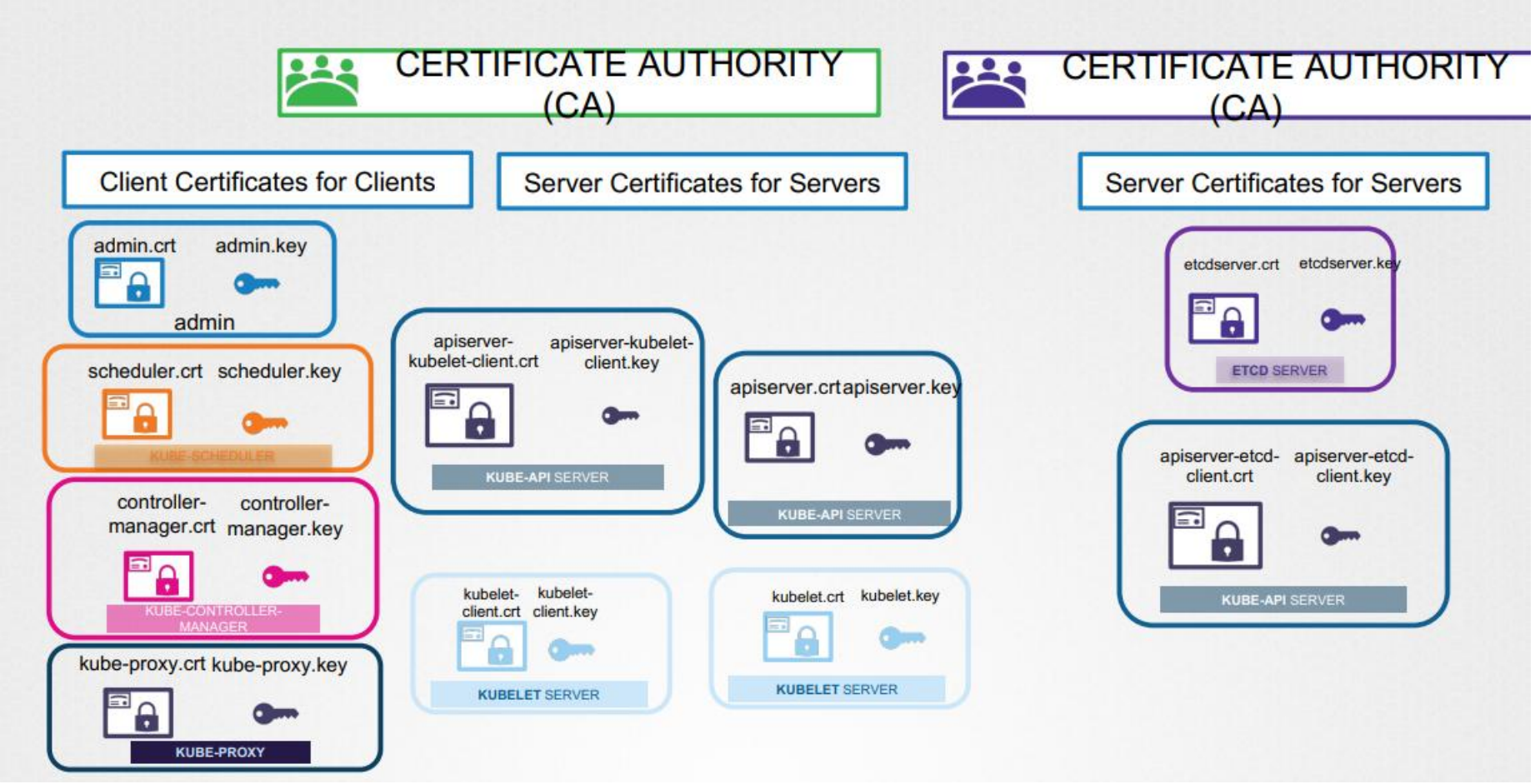
TLS Basics


Certificate (Public Key)
*.crt *.pem
server.crt
server.pem
client.crt
client.pem
Private Key
*.key *-key.pem
server.key
server-key.pem
client.key
client-key.pem
Public Key (Lock)
Private Key
User: John
Password:
Pass123
User: John
Password:
Pass123

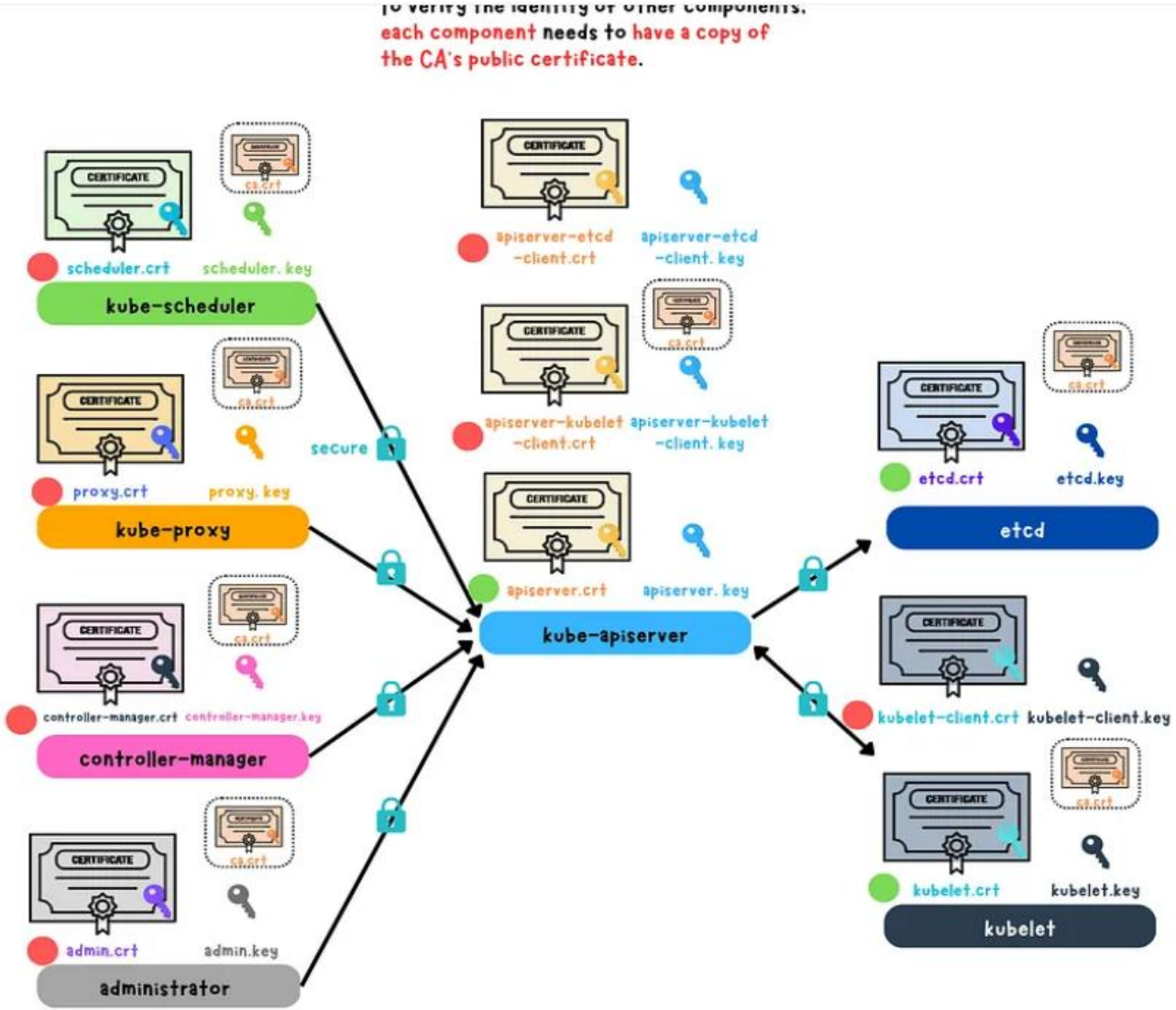
TLS in Kubernetes



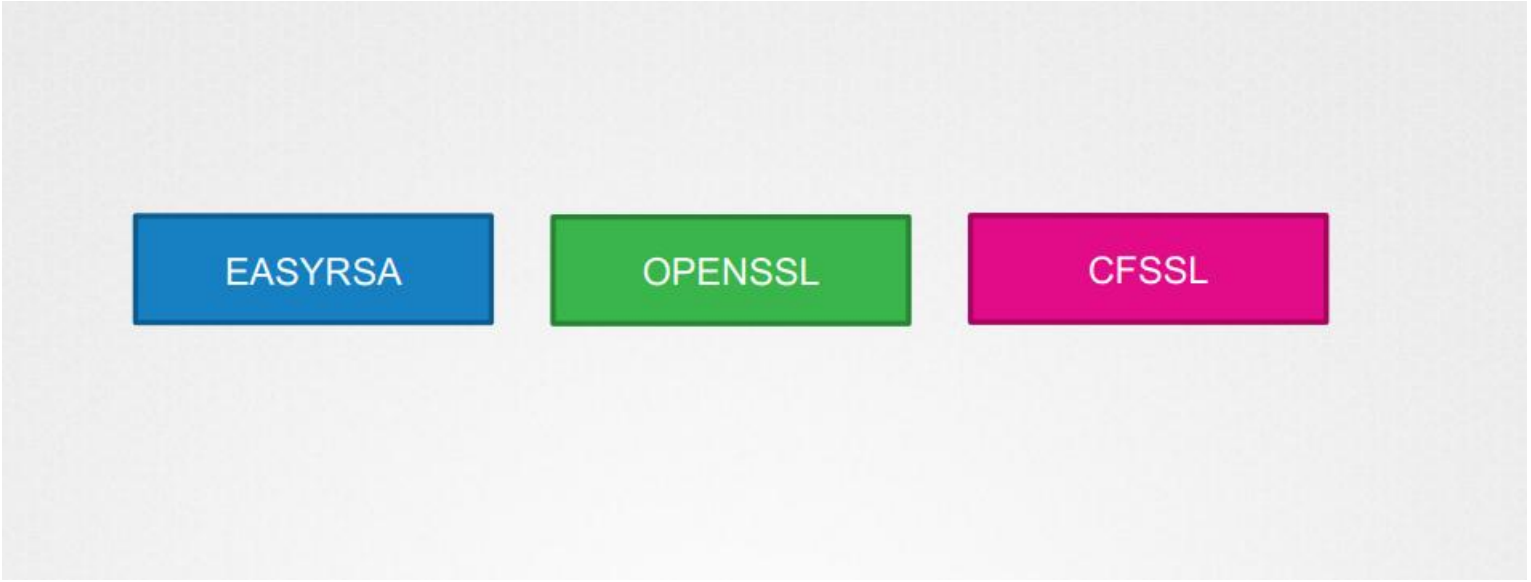
TLS in Kubernetes



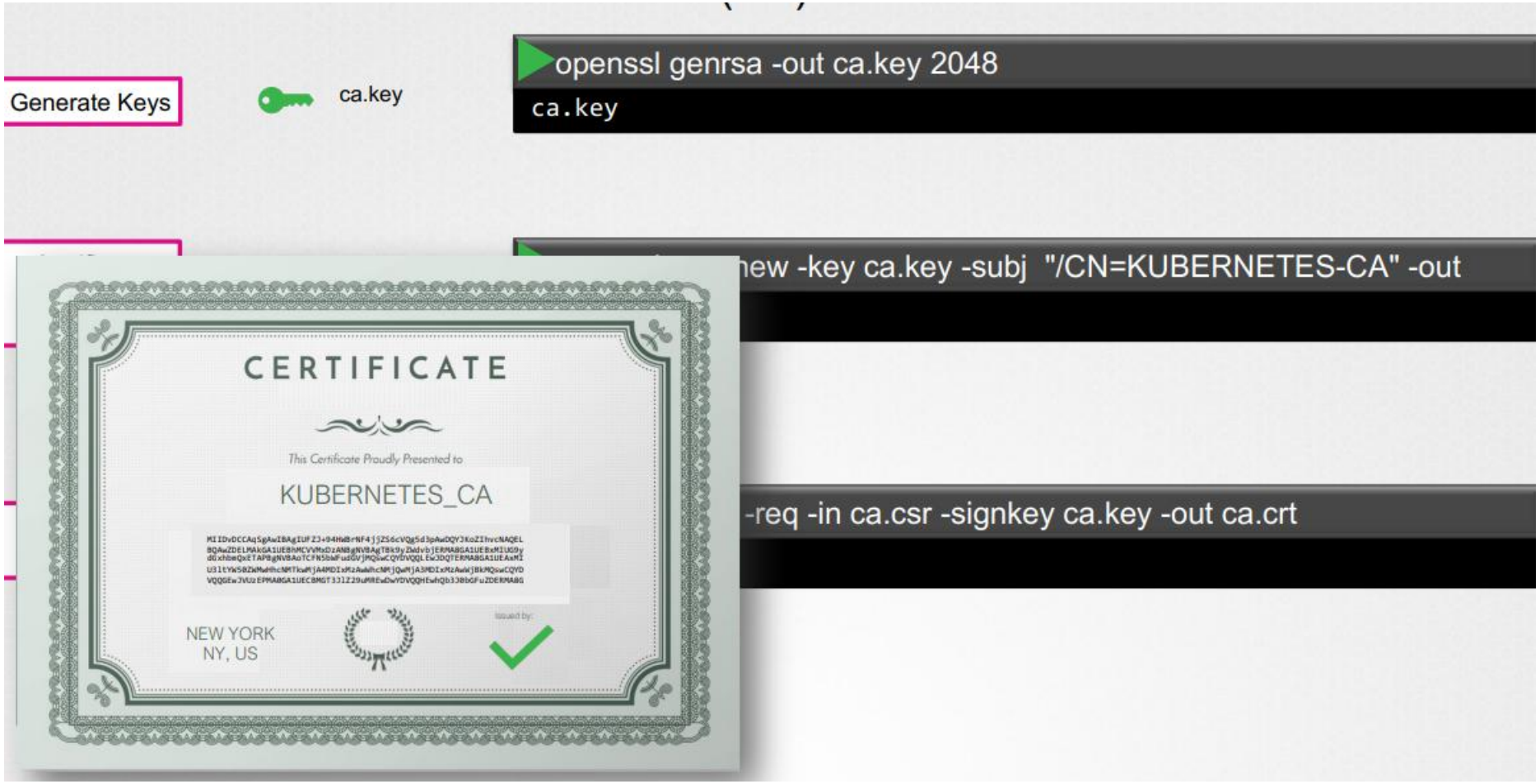
TLS in Kubernetes



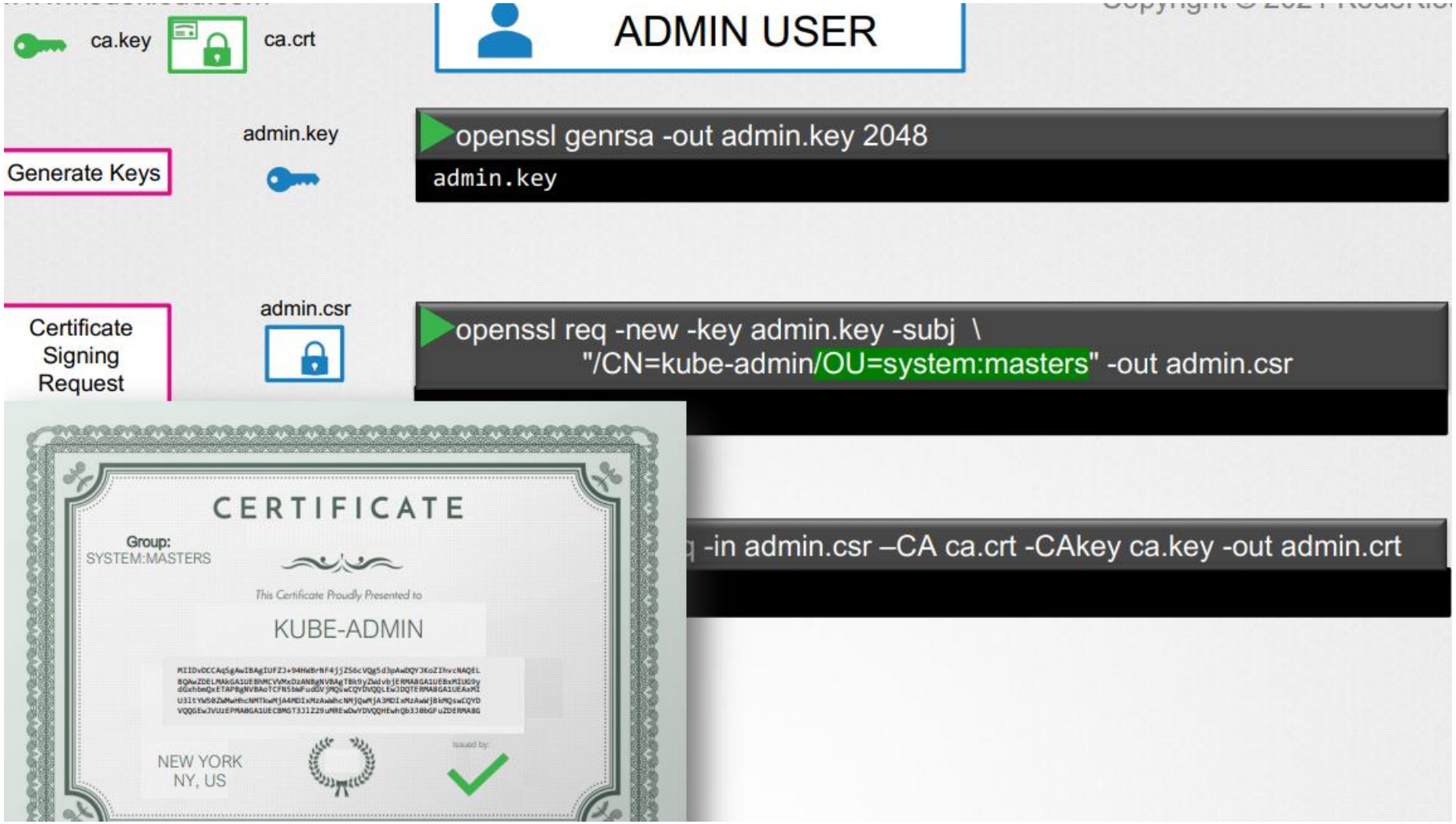
TLS in Kubernetes Generate Certificates



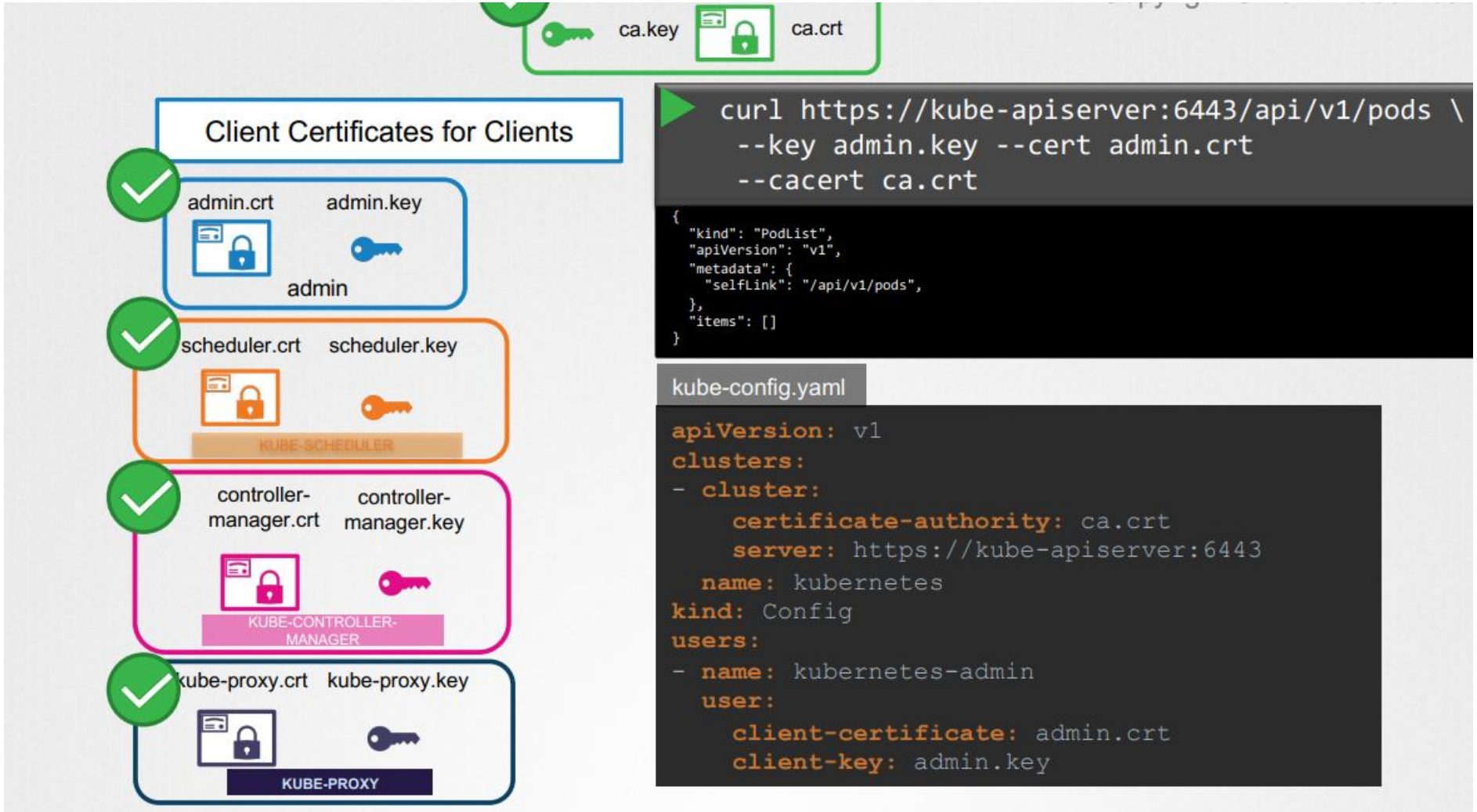
TLS in Kubernetes Generate Certificates



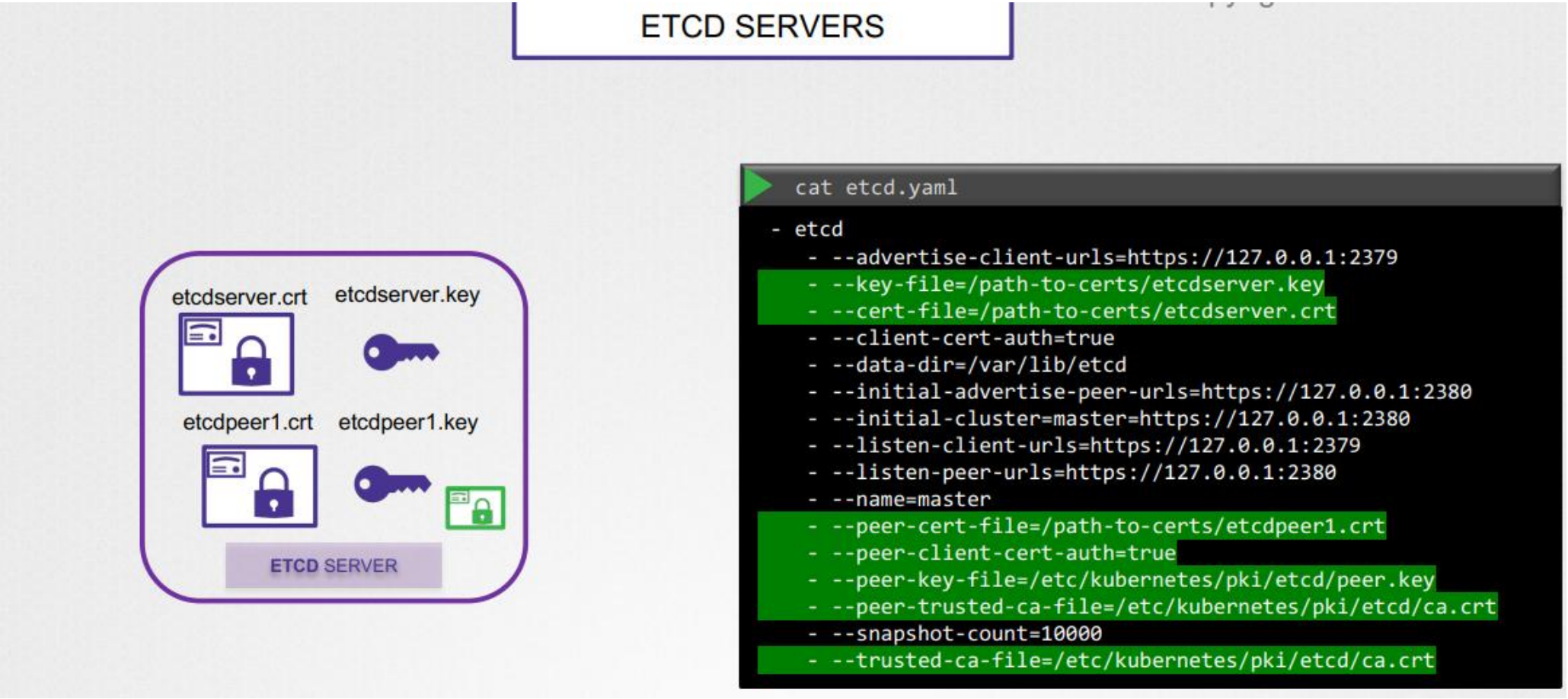
TLS in Kubernetes Generate Certificates



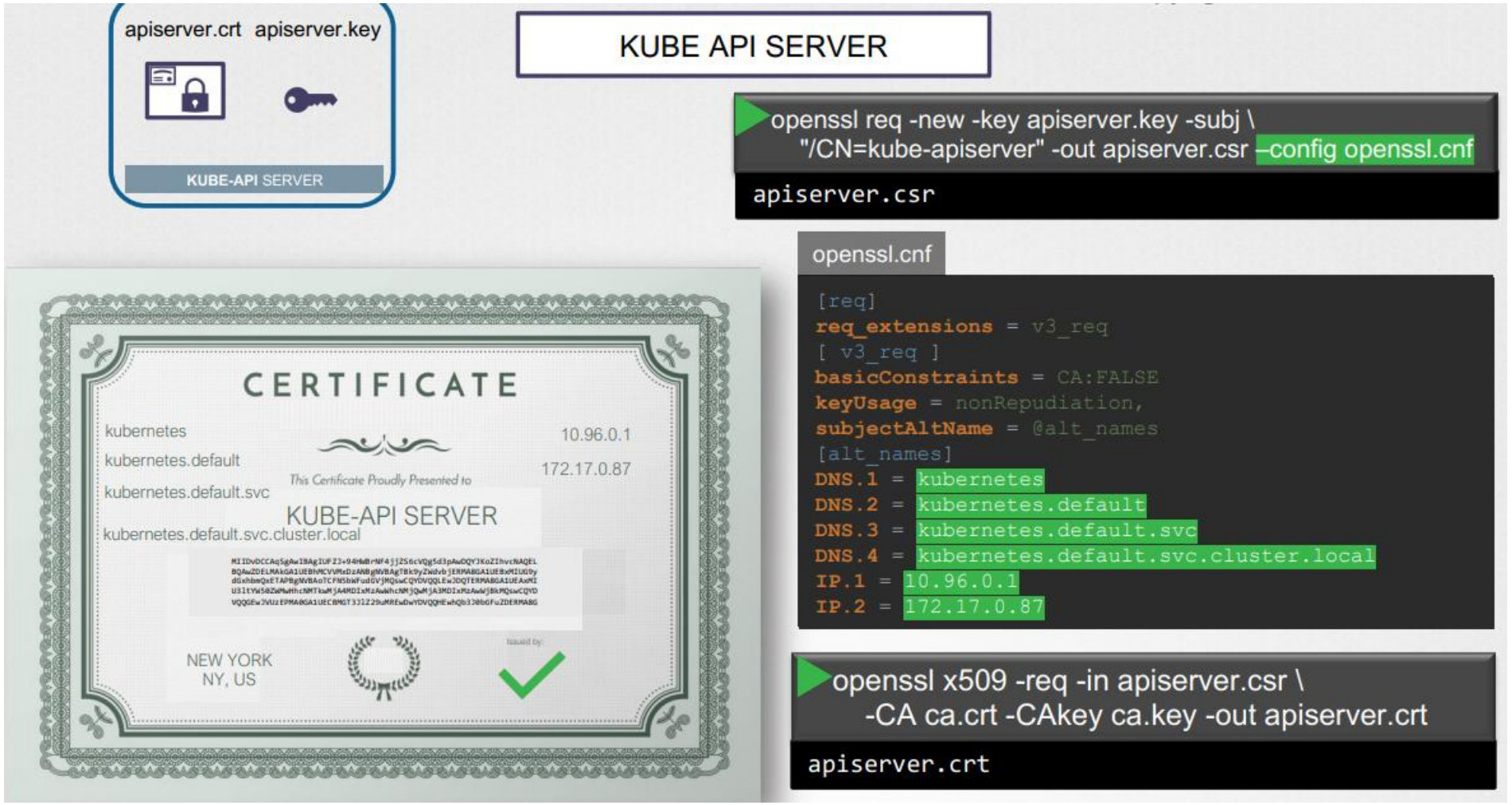
TLS in Kubernetes Generate Certificates



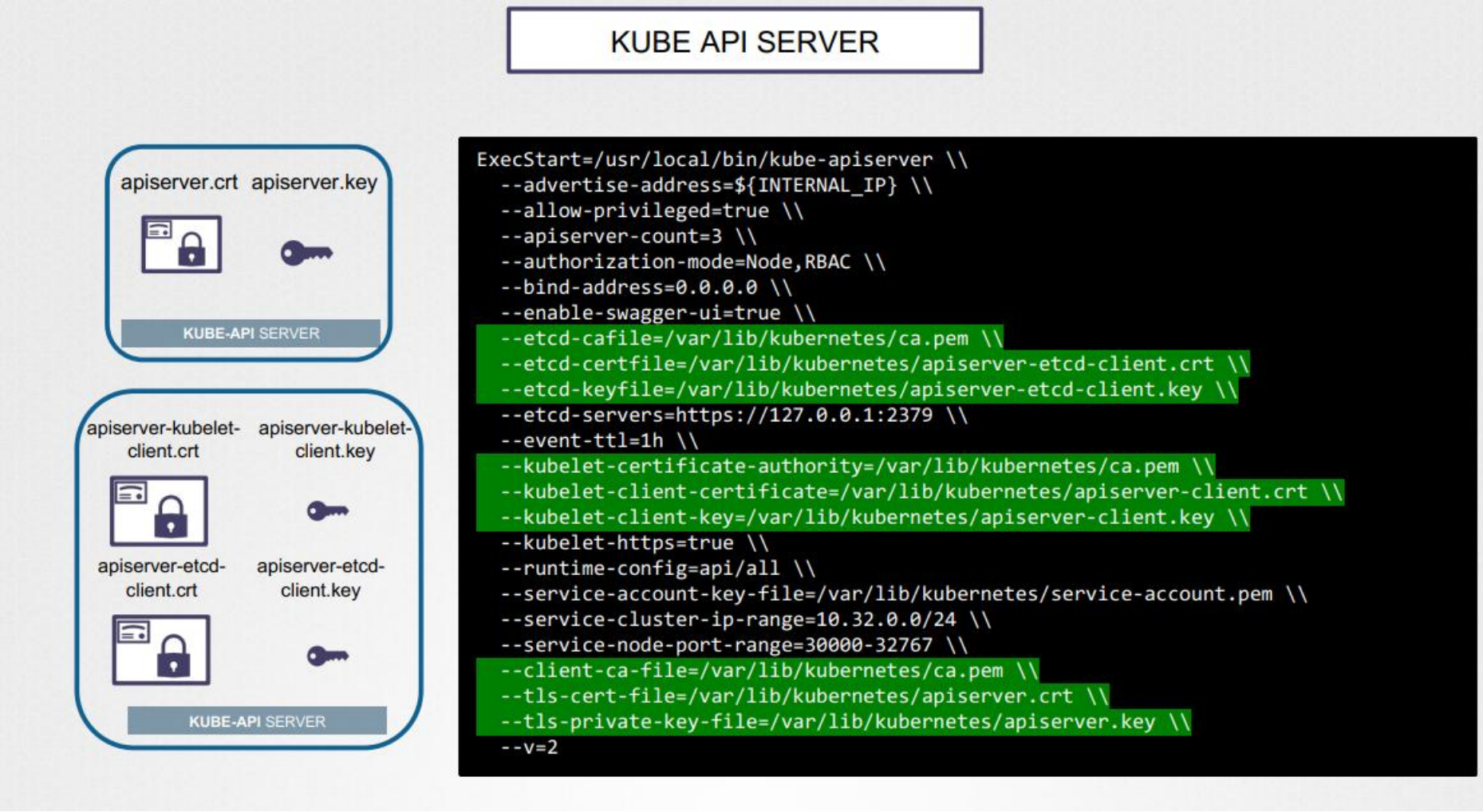
TLS in Kubernetes Generate Certificates



TLS in Kubernetes Generate Certificates



TLS in Kubernetes Generate Certificates



TLS in Kubernetes: View Certificate Details

/etc/kubernetes/pki/apiserver.crt

```
openssl x509 -in /etc/kubernetes/pki/apiserver.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3147495682089747350 (0x2bae26a58f090396)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=kubernetes
        Validity
            Not Before: Feb 11 05:39:19 2019 GMT
            Not After : Feb 11 05:39:20 2020 GMT
        Subject: CN=kube-apiserver
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:d9:69:38:80:68:3b:b7:2e:9e:25:00:e8:fd:01:

                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Alternative Name:
                DNS:master, DNS:kubernetes, DNS:kubernetes.default,
DNS:kubernetes.default.svc, DNS:kubernetes.default.svc.cluster.local, IP
Address:10.96.0.1, IP Address:172.17.0.27
```

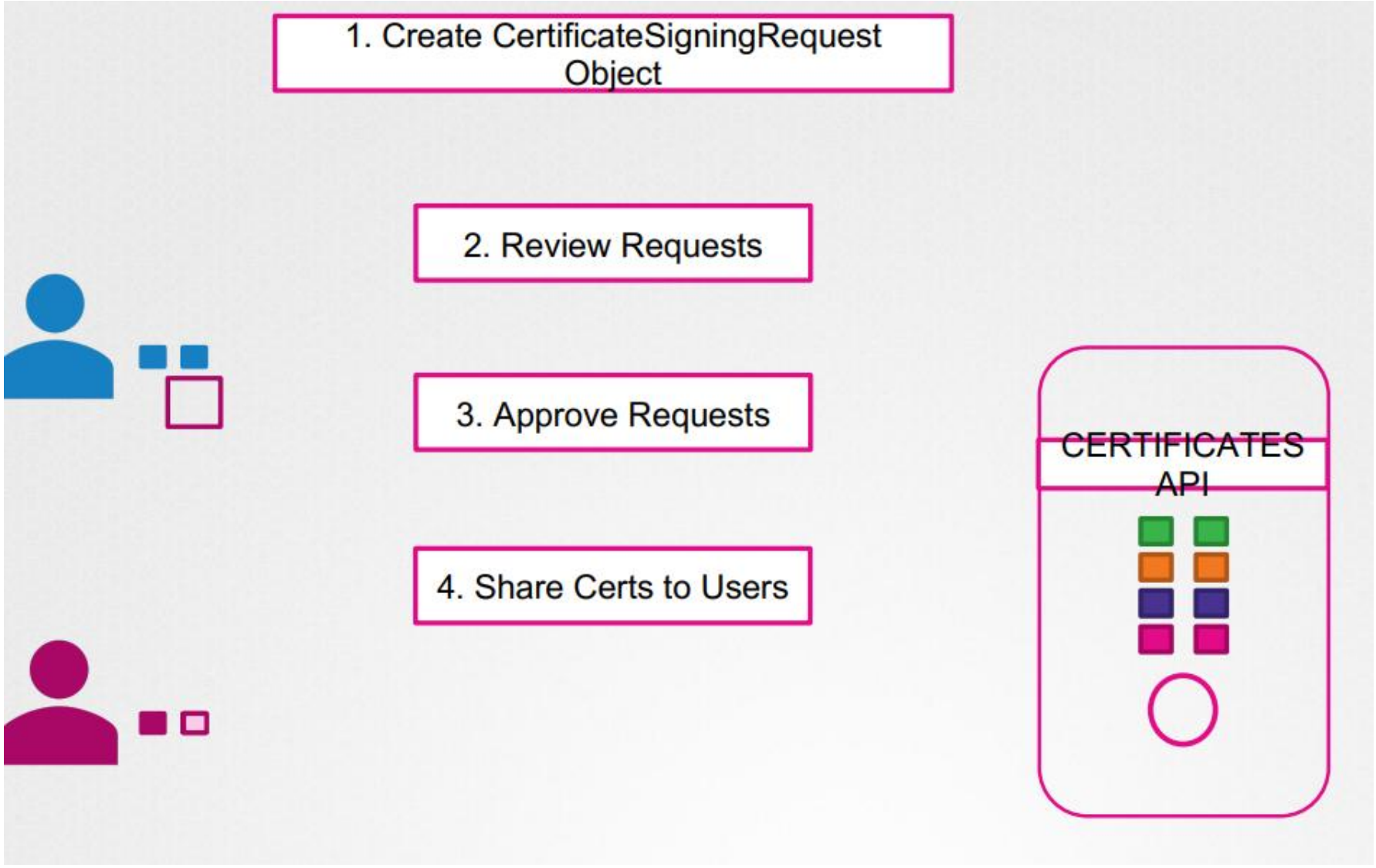
TLS in Kubernetes: View Certificate Details

Inspect Service Logs

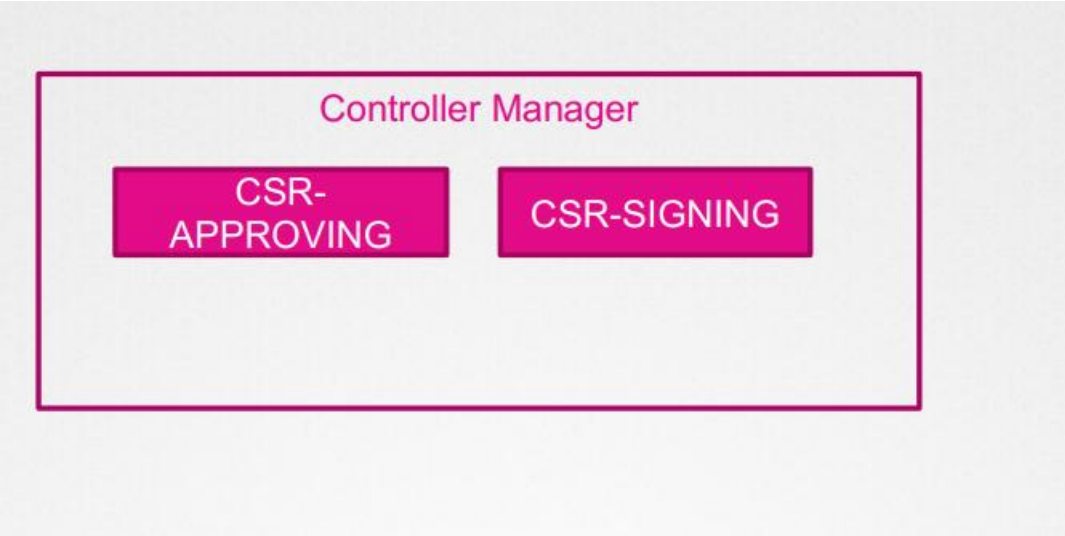
```
journalctl -u etcd.service -l
```

```
2019-02-13 02:53:28.144631 I | etcdmain: etcd Version: 3.2.18
2019-02-13 02:53:28.144680 I | etcdmain: Git SHA: eddf599c6
2019-02-13 02:53:28.144684 I | etcdmain: Go Version: go1.8.7
2019-02-13 02:53:28.144688 I | etcdmain: Go OS/Arch: linux/amd64
2019-02-13 02:53:28.144692 I | etcdmain: setting maximum number of CPUs to 4, total number of available CPUs is 4
2019-02-13 02:53:28.144734 N | etcdmain: the server is already initialized as member before, starting as etcd
member...
2019-02-13 02:53:28.146625 I | etcdserver: name = master
2019-02-13 02:53:28.146637 I | etcdserver: data dir = /var/lib/etcd
2019-02-13 02:53:28.146642 I | etcdserver: member dir = /var/lib/etcd/member
2019-02-13 02:53:28.146645 I | etcdserver: heartbeat = 100ms
2019-02-13 02:53:28.146648 I | etcdserver: election = 1000ms
2019-02-13 02:53:28.146651 I | etcdserver: snapshot count = 10000
2019-02-13 02:53:28.146677 I | etcdserver: advertise client URLs = 2019-02-13 02:53:28.185353 I | etcdserver/api:
enabled capabilities for version 3.2
2019-02-13 02:53:28.185588 I | embed: ClientTLS: cert = /etc/kubernetes/pki/etcd/server.crt, key =
/etc/kubernetes/pki/etcd/server.key, ca = , trusted-ca = /etc/kubernetes/pki/etcd/old-ca.crt, client-cert-auth =
true
2019-02-13 02:53:30.080017 I | embed: ready to serve client requests
2019-02-13 02:53:30.080130 I | etcdserver: published {Name:master ClientURLs:[https://127.0.0.1:2379]} to cluster
c9be114fc2da2776
2019-02-13 02:53:30.080281 I | embed: serving client requests on 127.0.0.1:2379
WARNING: 2019/02/13 02:53:30 Failed to dial 127.0.0.1:2379: connection error: desc = "transport: authentication
handshake failed: remote error: tls: bad certificate"; please retry.
```

Certificate API



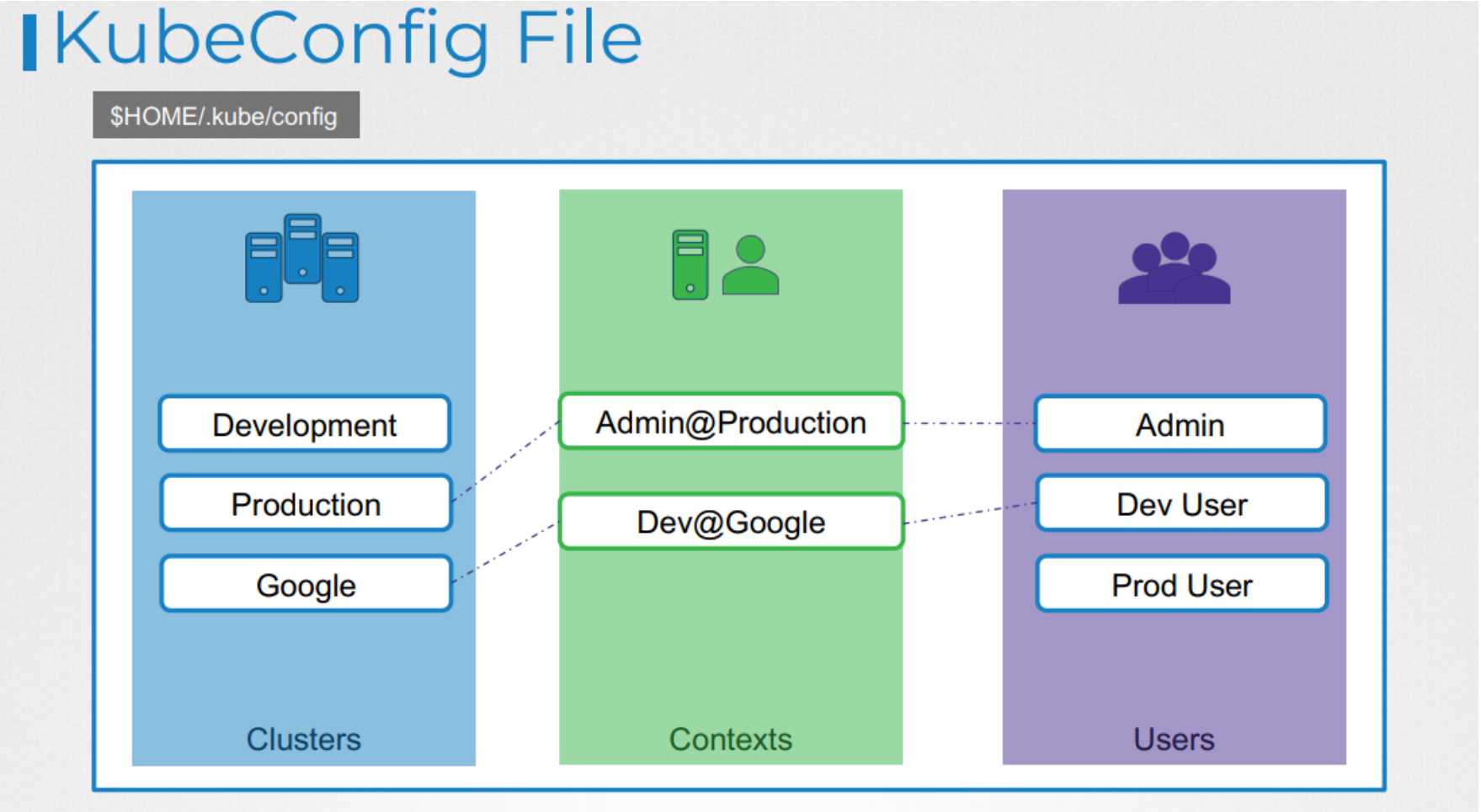
Certificate API



Certificate API

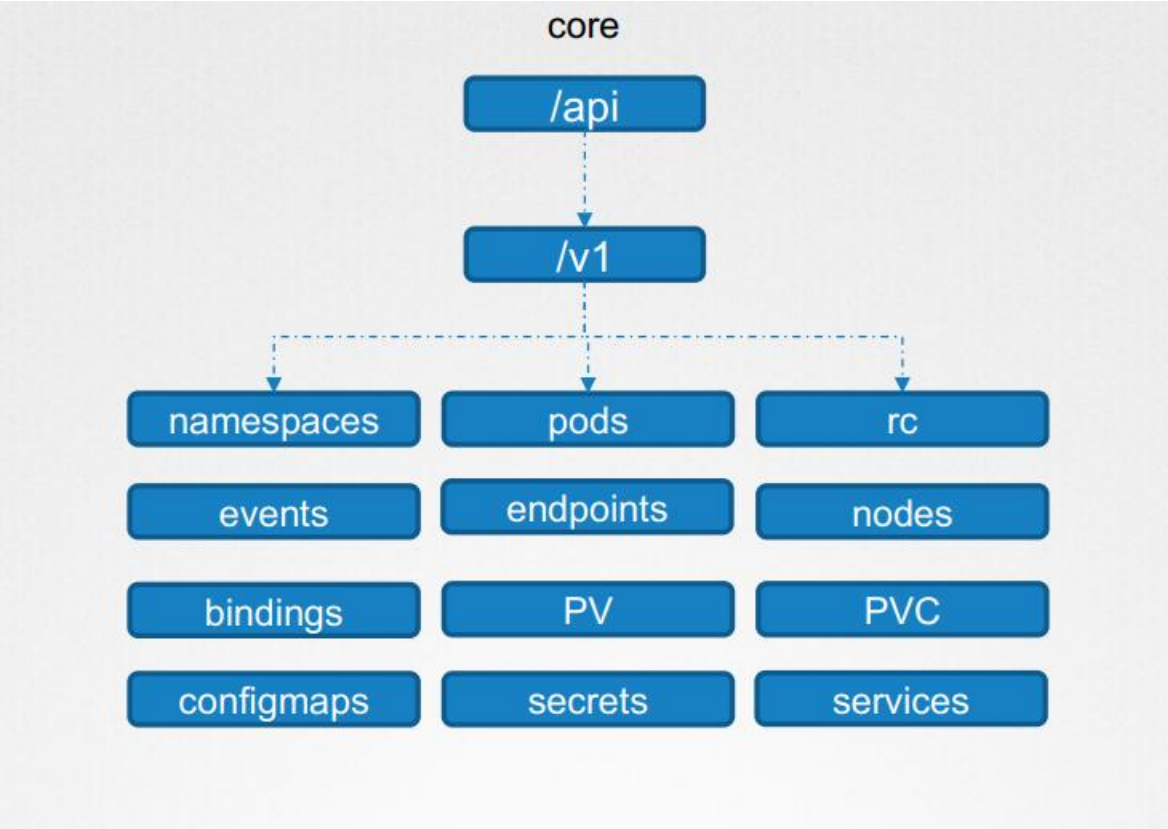
```
cat /etc/kubernetes/manifests/kube-controller-manager.yaml
spec:
  containers:
  - command:
    - kube-controller-manager
    - --address=127.0.0.1
    - --cluster-signing-cert-file=/etc/kubernetes/pki/ca.crt
    - --cluster-signing-key-file=/etc/kubernetes/pki/ca.key
    - --controllers=*,bootstrapsigner,tokencleaner
    - --kubeconfig=/etc/kubernetes/controller-manager.conf
    - --leader-elect=true
    - --root-ca-file=/etc/kubernetes/pki/ca.crt
    - --service-account-private-key-file=/etc/kubernetes/pki/sa.key
    - --use-service-account-credentials=true
```

Kube Config

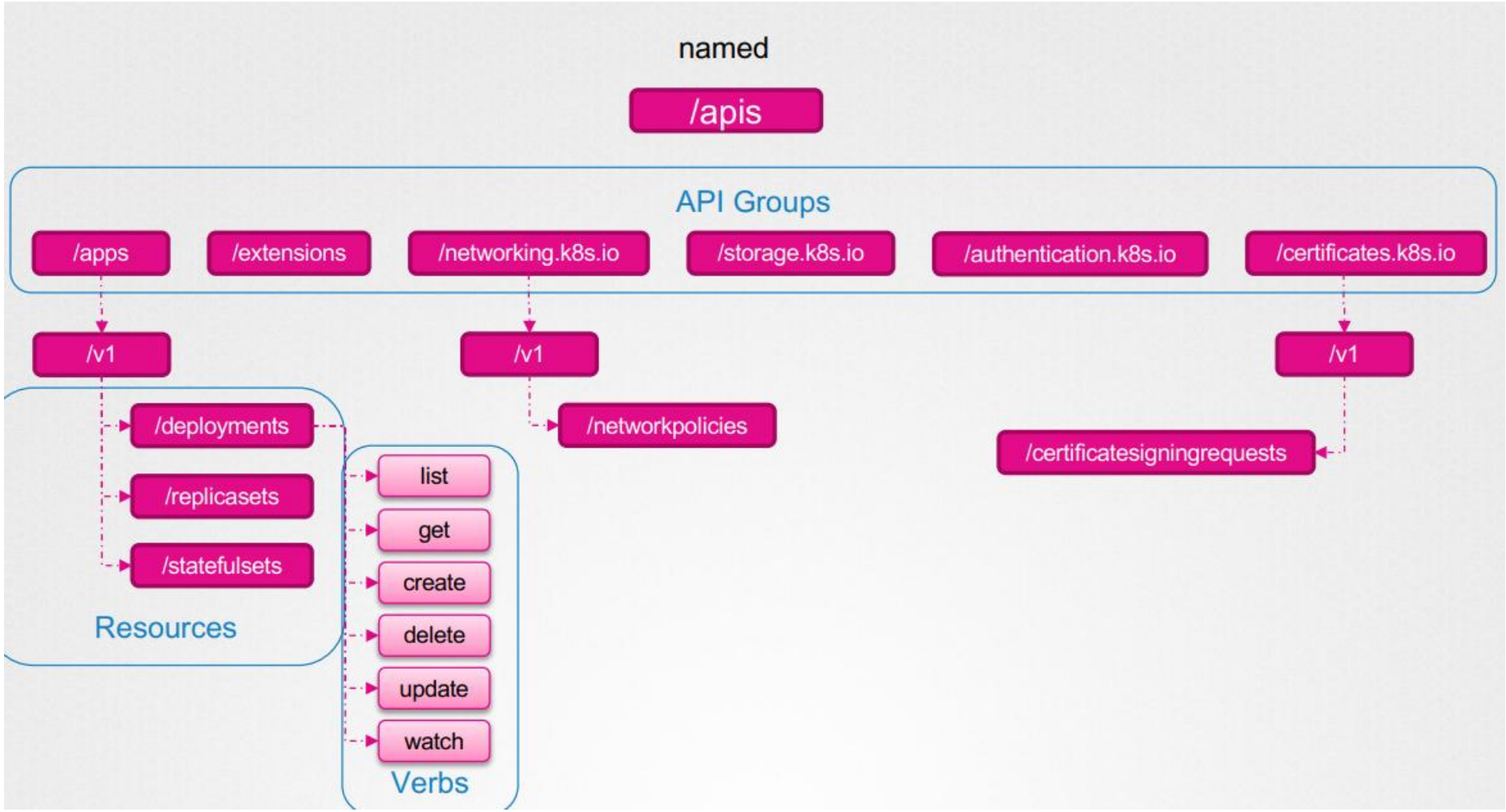

KubeConfig File
$HOME/.kube/config
Development
Production
Google
Admin@Production
Dev@Google
Admin
Dev User
Prod User
Clusters
Contexts
Users

Kubernetes API Groups

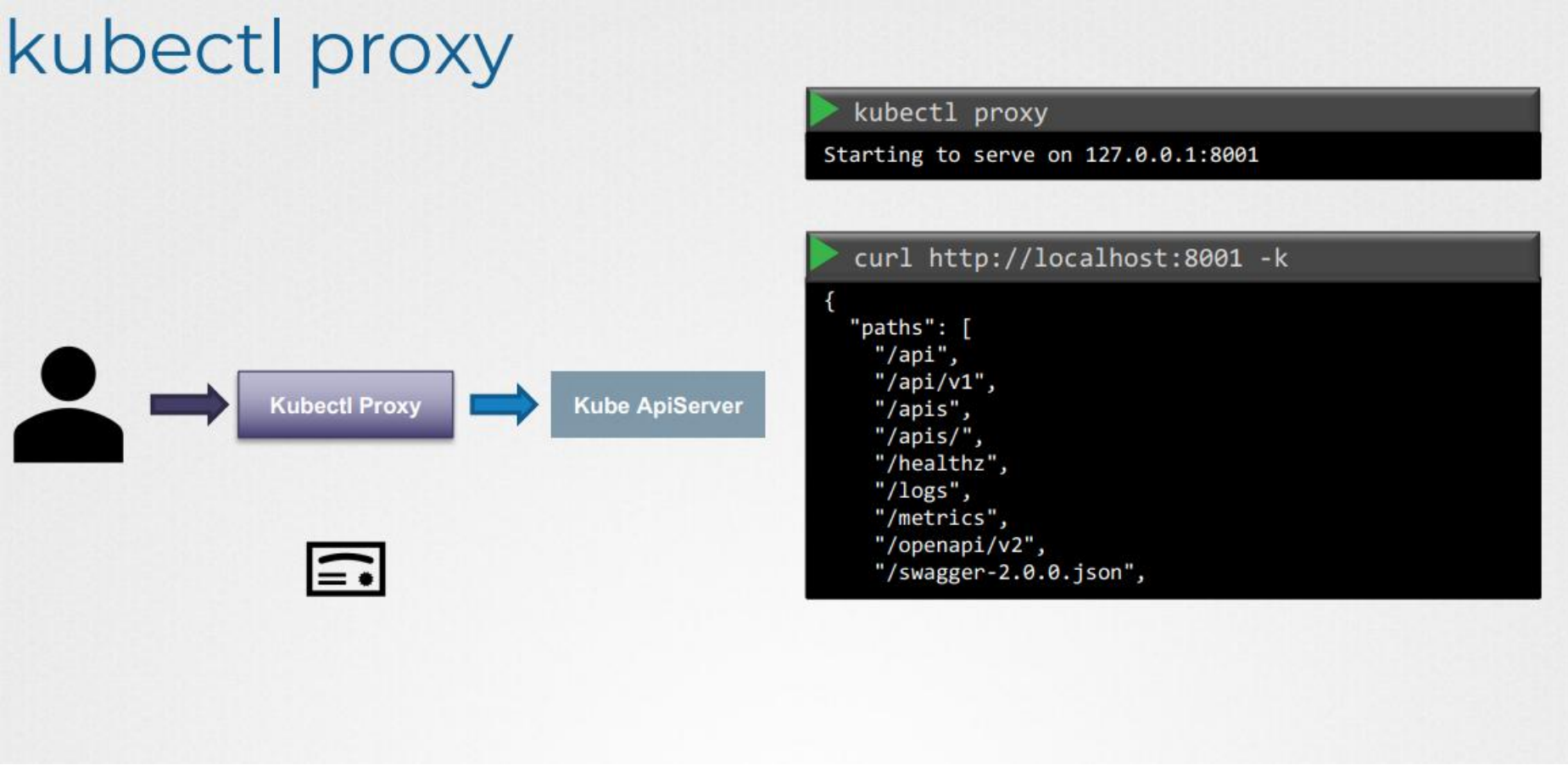
Kubernetes API Groups



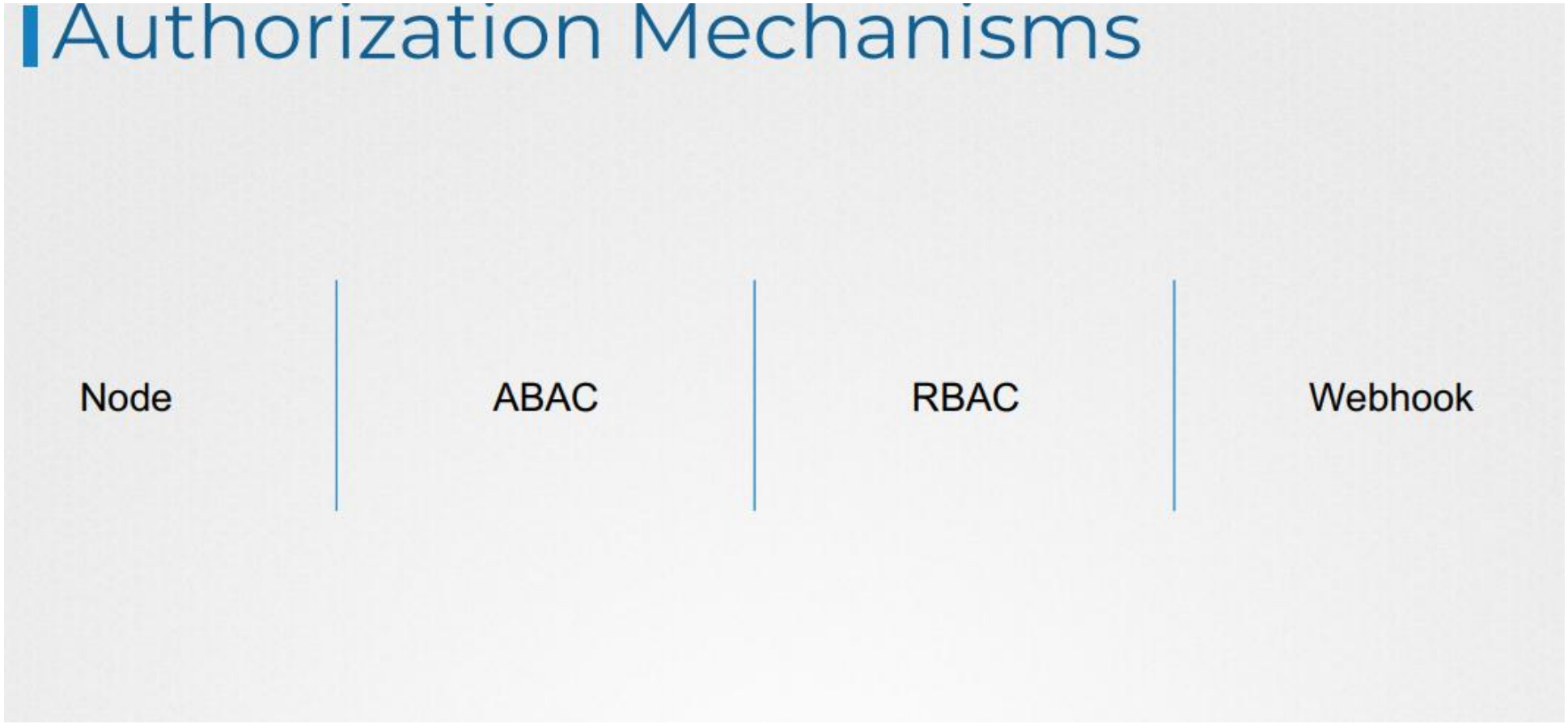
Kubernetes API Groups



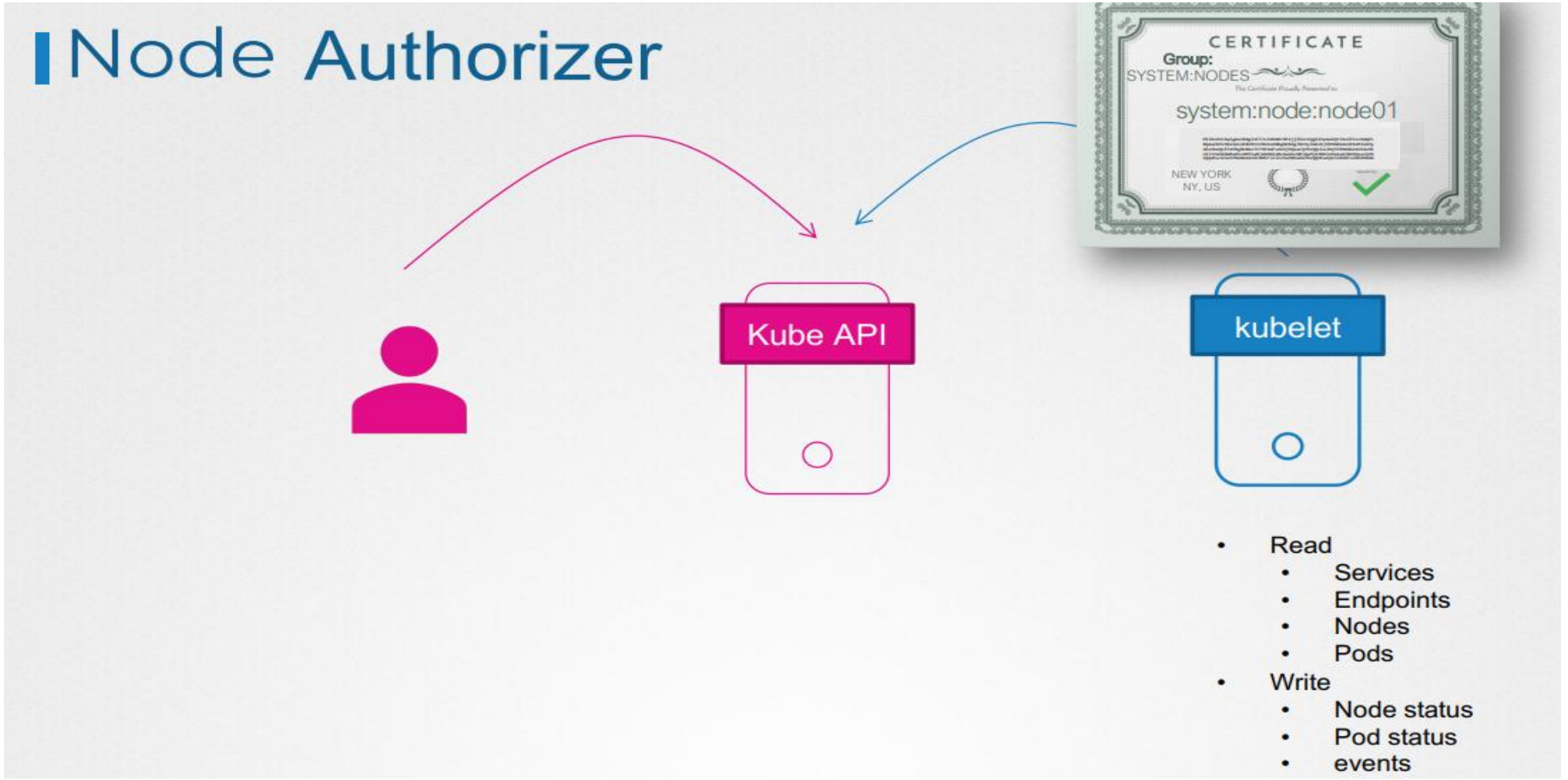
Kubernetes API Groups



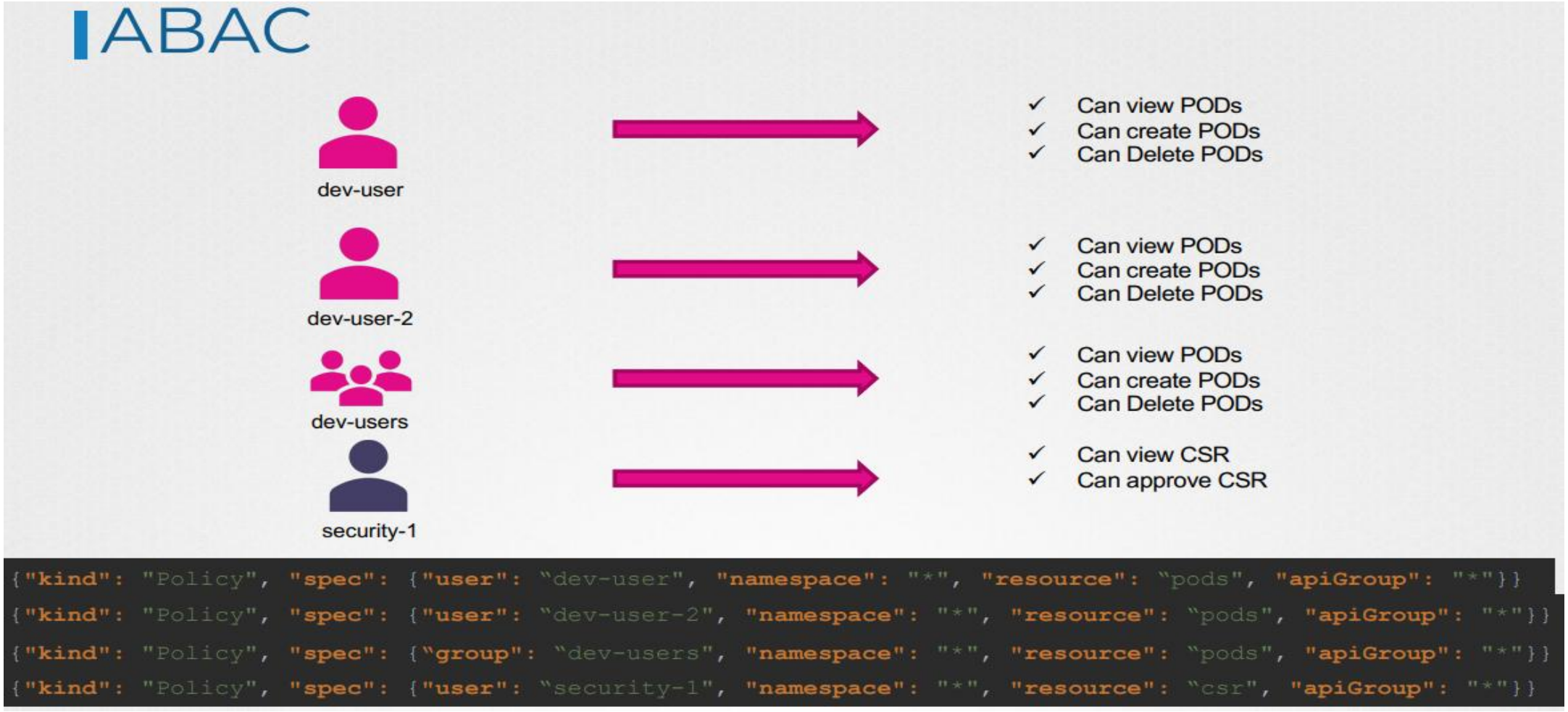
Authorization



Authorization

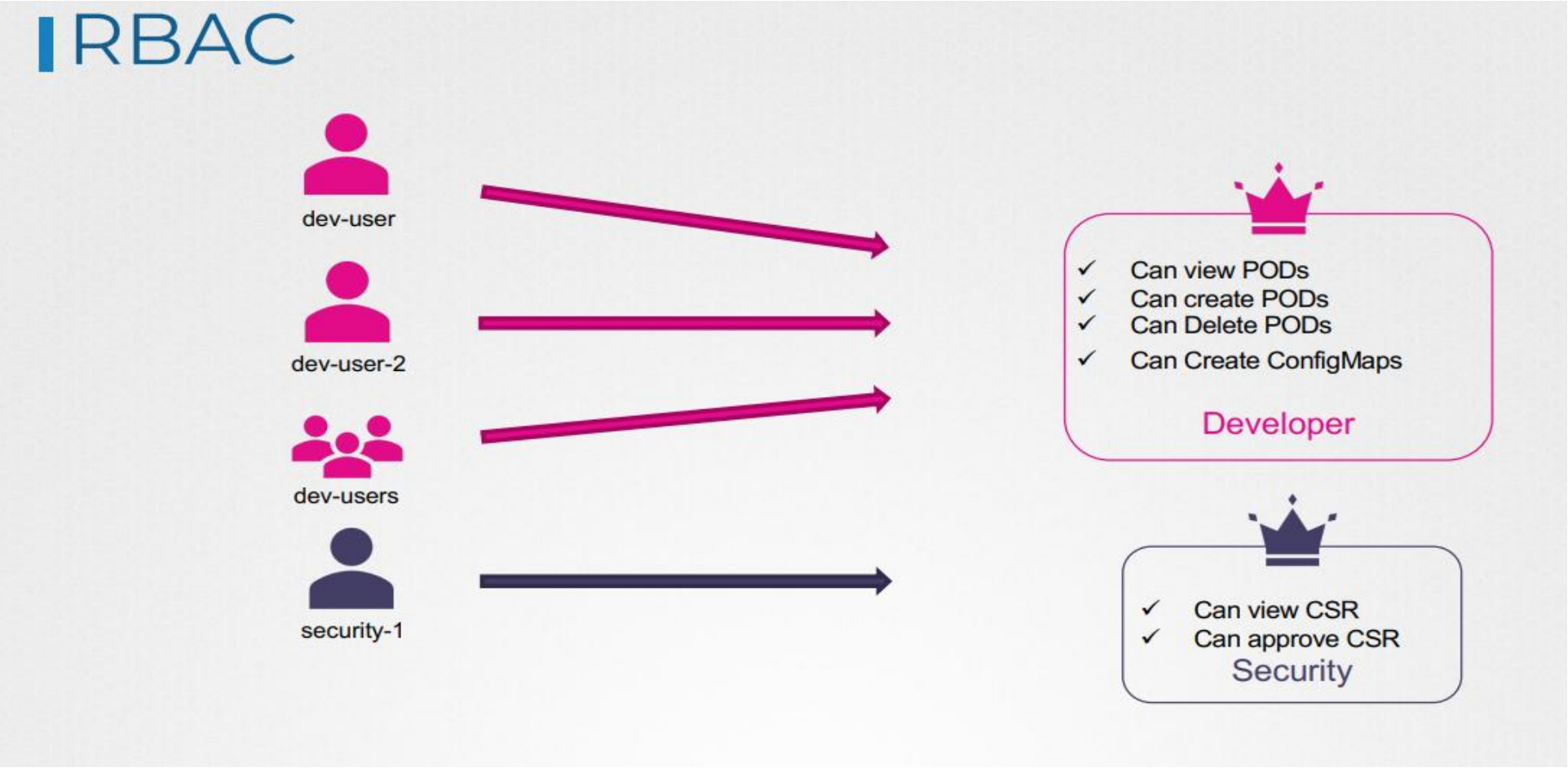


Authorization

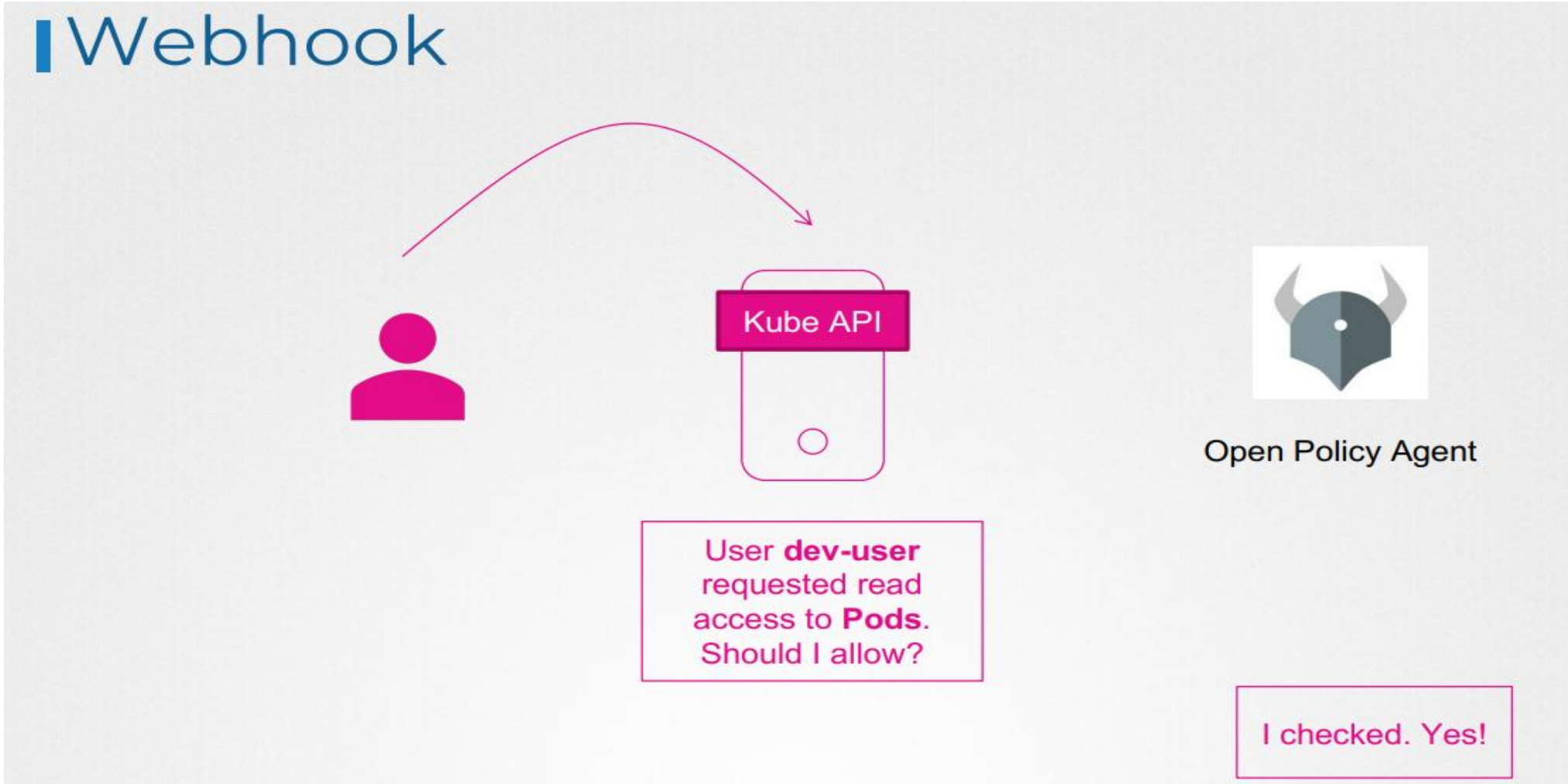



```
{"kind": "Policy", "spec": {"user": "dev-user", "namespace": "*", "resource": "pods", "apiGroup": "*"}}
{"kind": "Policy", "spec": {"user": "dev-user-2", "namespace": "*", "resource": "pods", "apiGroup": "*"}}
{"kind": "Policy", "spec": {"group": "dev-users", "namespace": "*", "resource": "pods", "apiGroup": "*"}}
{"kind": "Policy", "spec": {"user": "security-1", "namespace": "*", "resource": "csr", "apiGroup": "*"}}
```

Authorization



Authorization



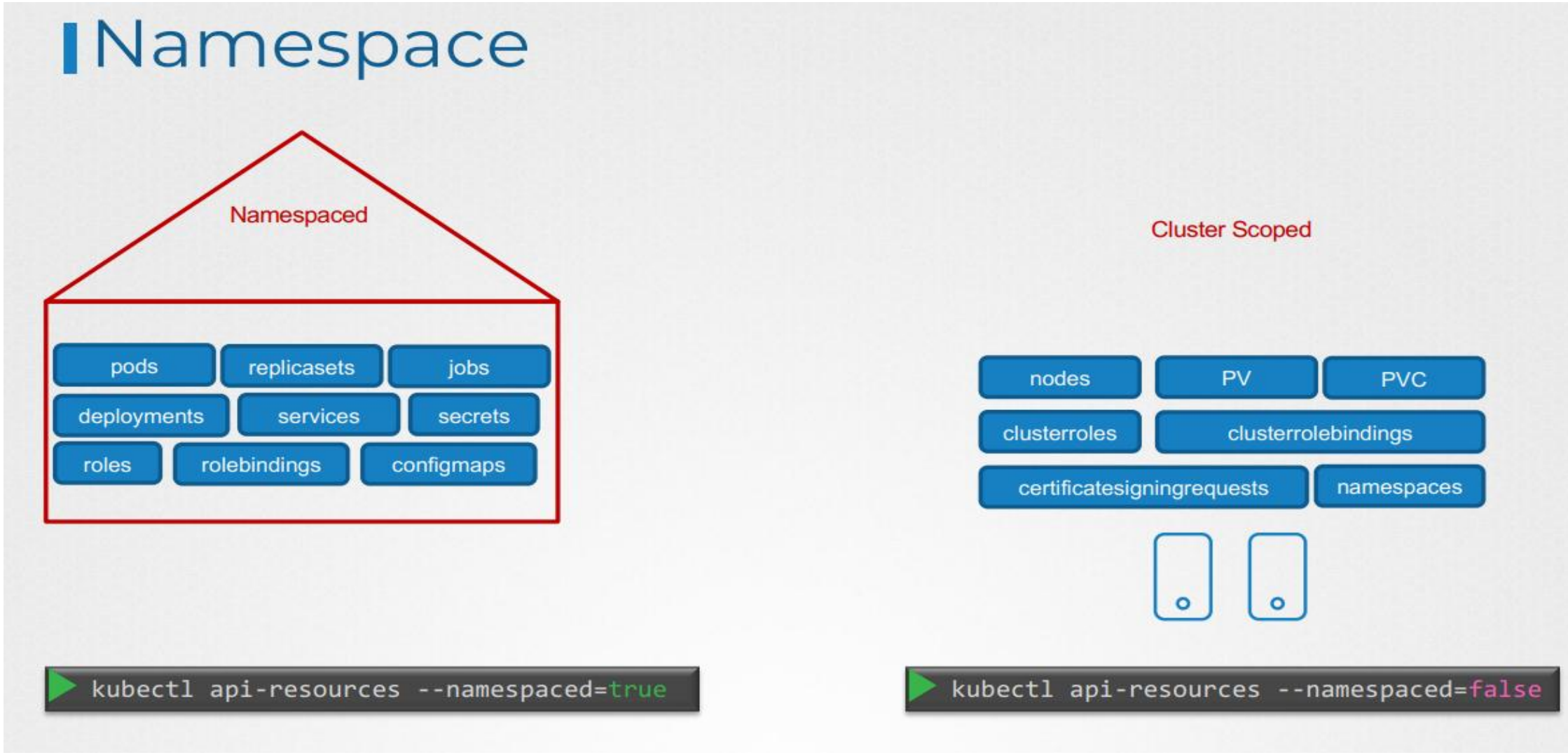
Authorization

Authorization Mode

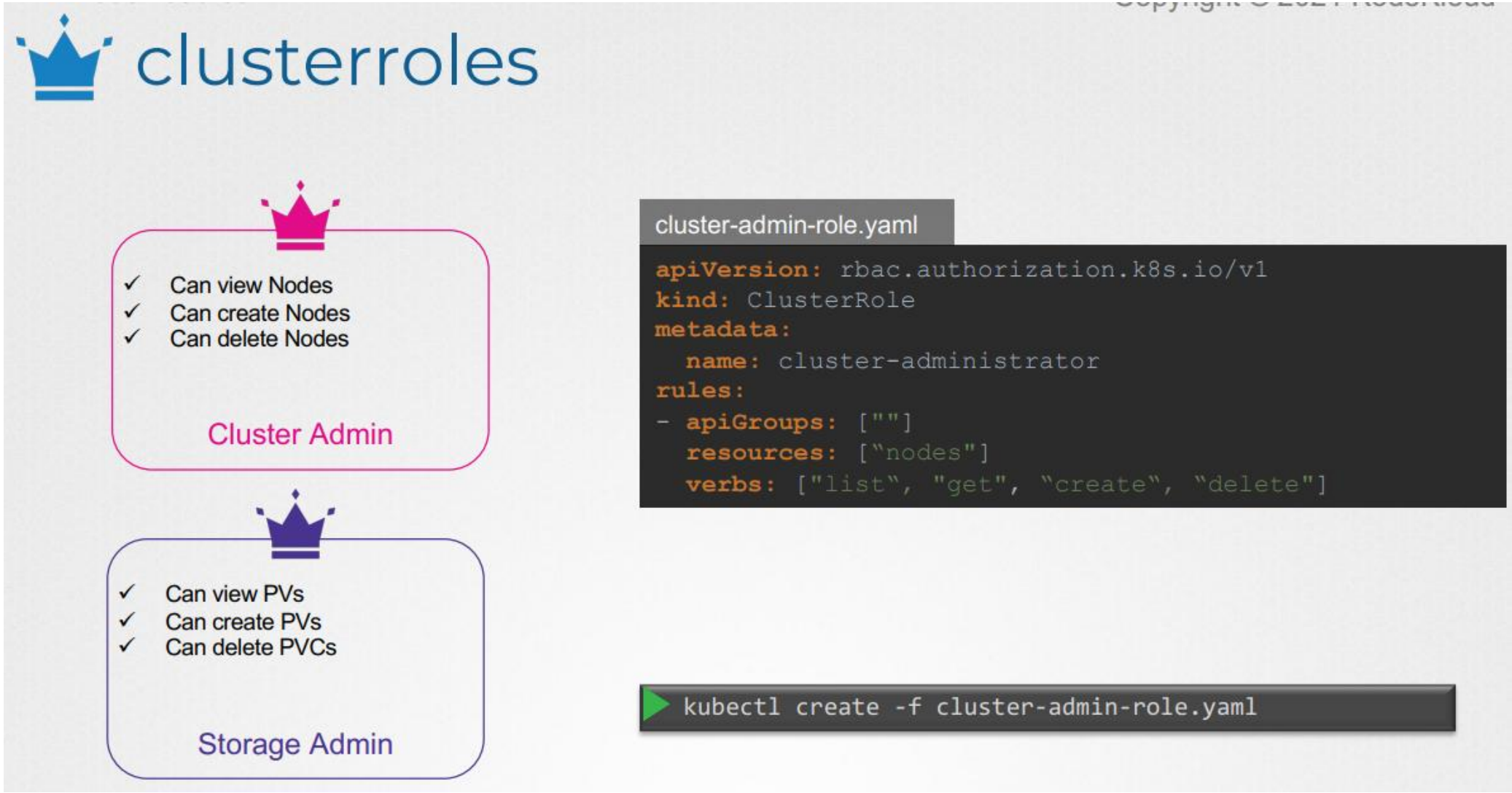
AlwaysAllow | NODE | ABAC | RBAC | WEBHOOK | AlwaysDeny

```
ExecStart=/usr/local/bin/kube-apiserver \\
  --advertise-address=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --authorization-mode=Node,RBAC,Webhook \\
  --bind-address=0.0.0.0 \\
  --enable-swagger-ui=true \\
  --etcd-cafile=/var/lib/kubernetes/ca.pem \\
  --etcd-certfile=/var/lib/kubernetes/apiserver-etcd-client.crt \\
  --etcd-keyfile=/var/lib/kubernetes/apiserver-etcd-client.key \\
  --etcd-servers=https://127.0.0.1:2379 \\
  --event-ttl=1h \\
  --kubelet-certificate-authority=/var/lib/kubernetes/ca.pem \\
  --kubelet-client-certificate=/var/lib/kubernetes/apiserver-etcd-client.crt \\
  --kubelet-client-key=/var/lib/kubernetes/apiserver-etcd-client.key \\
  --service-node-port-range=30000-32767 \\
  --client-ca-file=/var/lib/kubernetes/ca.pem \\
  --tls-cert-file=/var/lib/kubernetes/apiserver.crt \\
  --tls-private-key-file=/var/lib/kubernetes/apiserver.key \\
  --v=2
```

Cluster Roles and Bindings



Cluster Roles and Bindings



Kubelet Security

Installing kubelet

```
wget https://storage.googleapis.com/kubernetes-release/release/v1.20.0/bin/linux/amd64/kubelet
```

kubelet.service

```
ExecStart=/usr/local/bin/kubelet \\
  --container-runtime=docker \\
  --image-pull-progress-deadline=2m \\
  --kubeconfig=/var/lib/kubelet/kubeconfig \\
  --network-plugin=cni \\
  --register-node=true \\
  --v=2 \\
  --cluster-domain=cluster.local \\
  --file-check-frequency=0s \\
  --healthz-port=10248 \\
  --cluster-dns=10.96.0.10 \\
  --http-check-frequency=0s \\
  --sync-frequency=0s
```

Kubeadm does not deploy Kubelets

Kubelet Security

View kubelet options

```
ps -aux | grep kubelet
```

```
root      2095  1.8  2.4 960676 98788 ?        Ssl  02:32   0:36 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --config=/var/lib/kubelet/config.yaml --cgroup-driver=cgroupfs --cni-bin-dir=/opt/cni/bin --cni-conf-dir=/etc/cni/net.d --network-plugin=cni
```

```
cat /var/lib/kubelet/config.yaml
```

```
apiVersion: kubelet.config.k8s.io/v1beta1
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
rotateCertificates: true
runtimeRequestTimeout: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
```

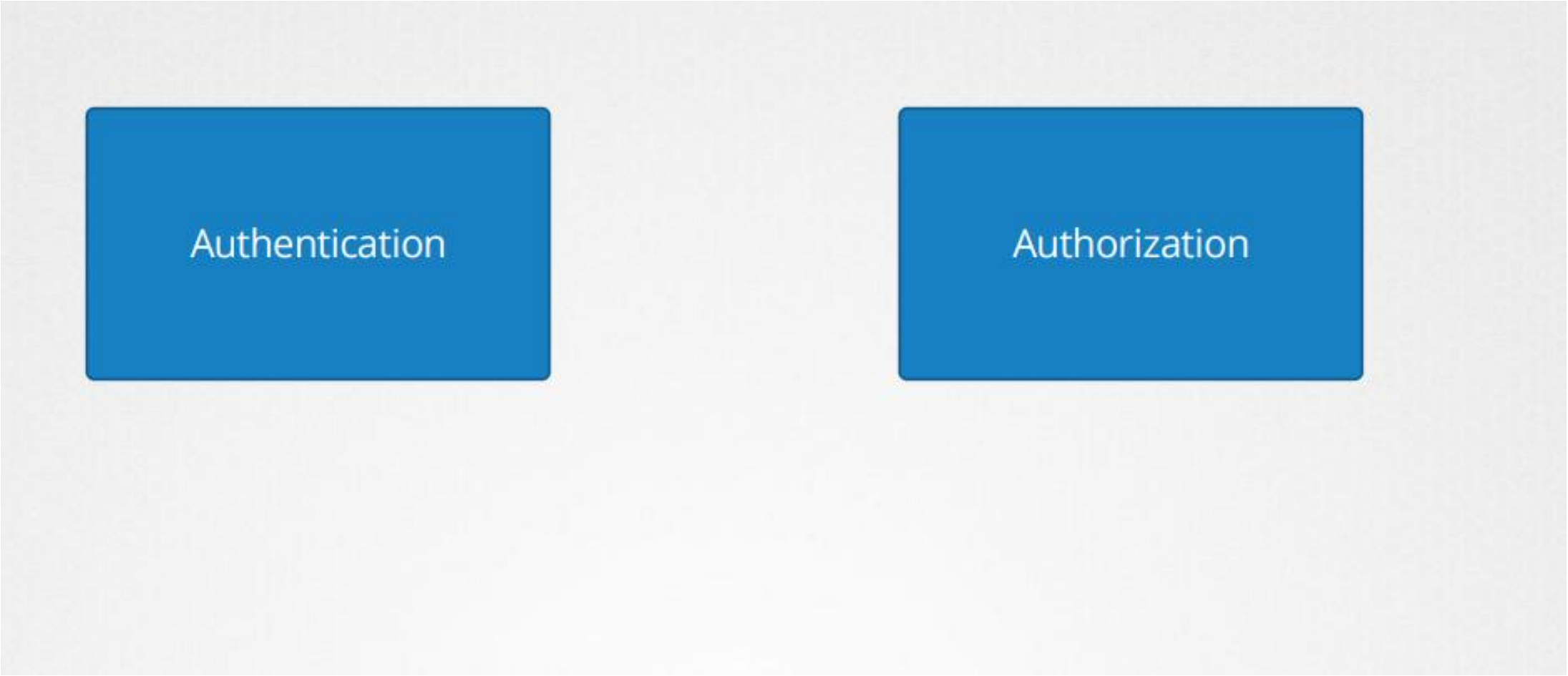
Kubelet Security

Kubelet

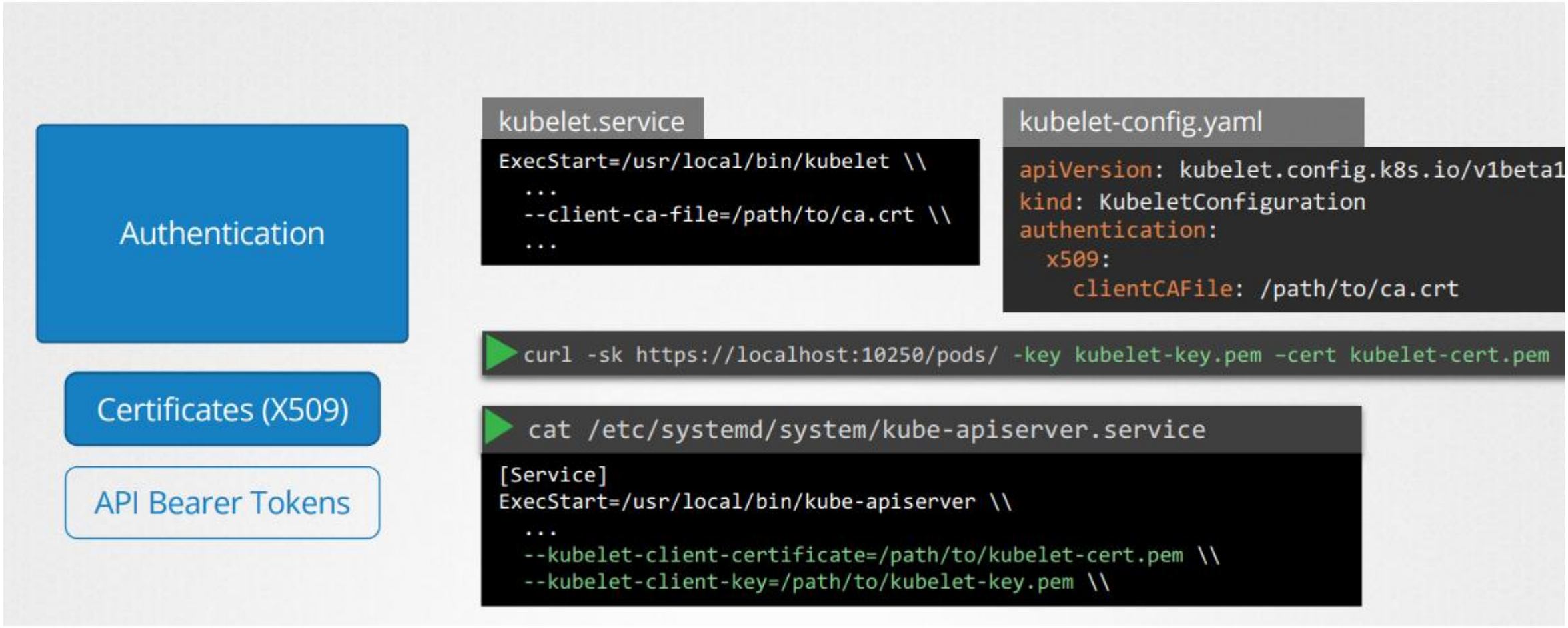
Port	Description
10250	Serves API that allows full access
10255	Serves API that allows unauthentcated read-only access

```
curl -sk http://localhost:10255/metrics
# HELP apiserver_audit_event_total [ALPHA] Counter of audit events generated and sent to the audit backend.
# TYPE apiserver_audit_event_total counter
apiserver_audit_event_total 0
# HELP apiserver_audit_requests_rejected_total [ALPHA] Counter of apiserver requests rejected due to an error in audit l
# TYPE apiserver_audit_requests_rejected_total counter
apiserver_audit_requests_rejected_total 0
# HELP apiserver_client_certificate_expiration_seconds [ALPHA] Distribution of the remaining lifetime on the certificate
requ
est.
# TYPE apiserver_client_certificate_expiration_seconds histogram
apiserver_client_certificate_expiration_seconds_bucket{le="0"} 0
apiserver_client_certificate_expiration_seconds_bucket{le="1800"} 0
apiserver_client_certificate_expiration_seconds_bucket{le="3600"} 0
apiserver_client_certificate_expiration_seconds_bucket{le="7200"} 0
apiserver_client_certificate_expiration_seconds_bucket{le="21600"} 0
apiserver_client_certificate_expiration_seconds_bucket{le="43200"} 0
apiserver_client_certificate_expiration_seconds_bucket{le="86400"} 0
apiserver_client_certificate_expiration_seconds_bucket{le="172800"} 0
```

Kubelet Security



Kubelet Security



Kubelet Security

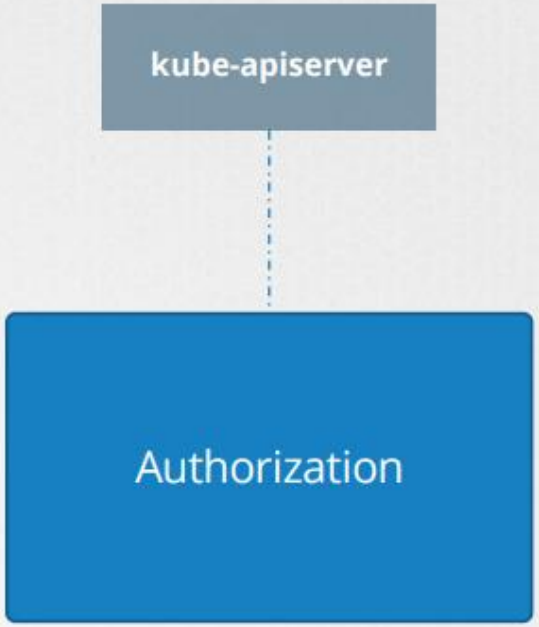
Kubelet Security

kubelet.service

```
ExecStart=/usr/local/bin/kubelet \\
  ...
  --authorization-mode=Webhook
  ...
```

kubelet-config.yaml

```yaml
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
authorization:
  mode: Webhook
```



Kubelet Security

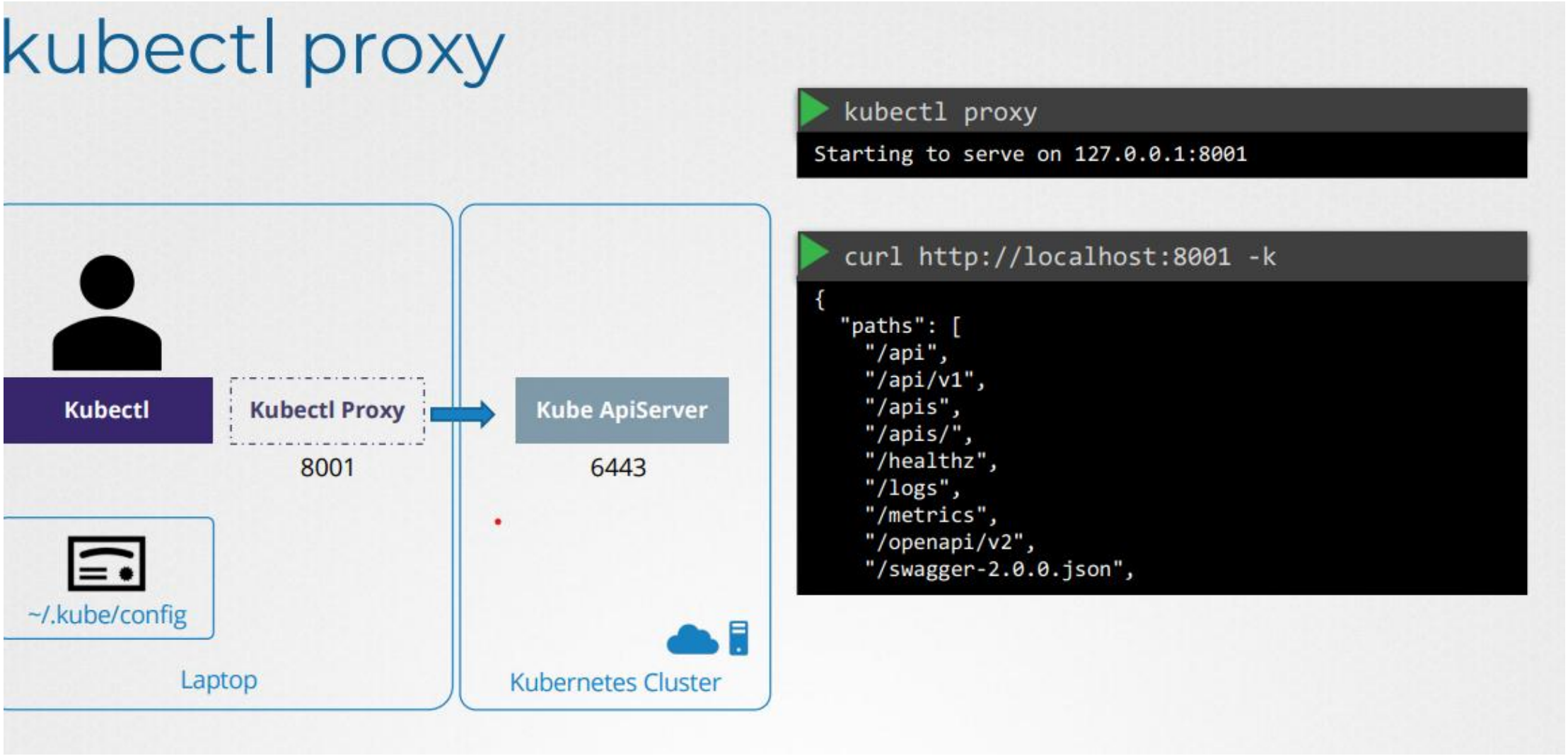
kubelet.service

```
ExecStart=/usr/local/bin/kubelet \\
  ...
  --anonymous-auth=false \\
  --client-ca-file=/path/to/ca.crt \\
  --authorization-mode=Webhook
  --read-only-port=0
  ...
```

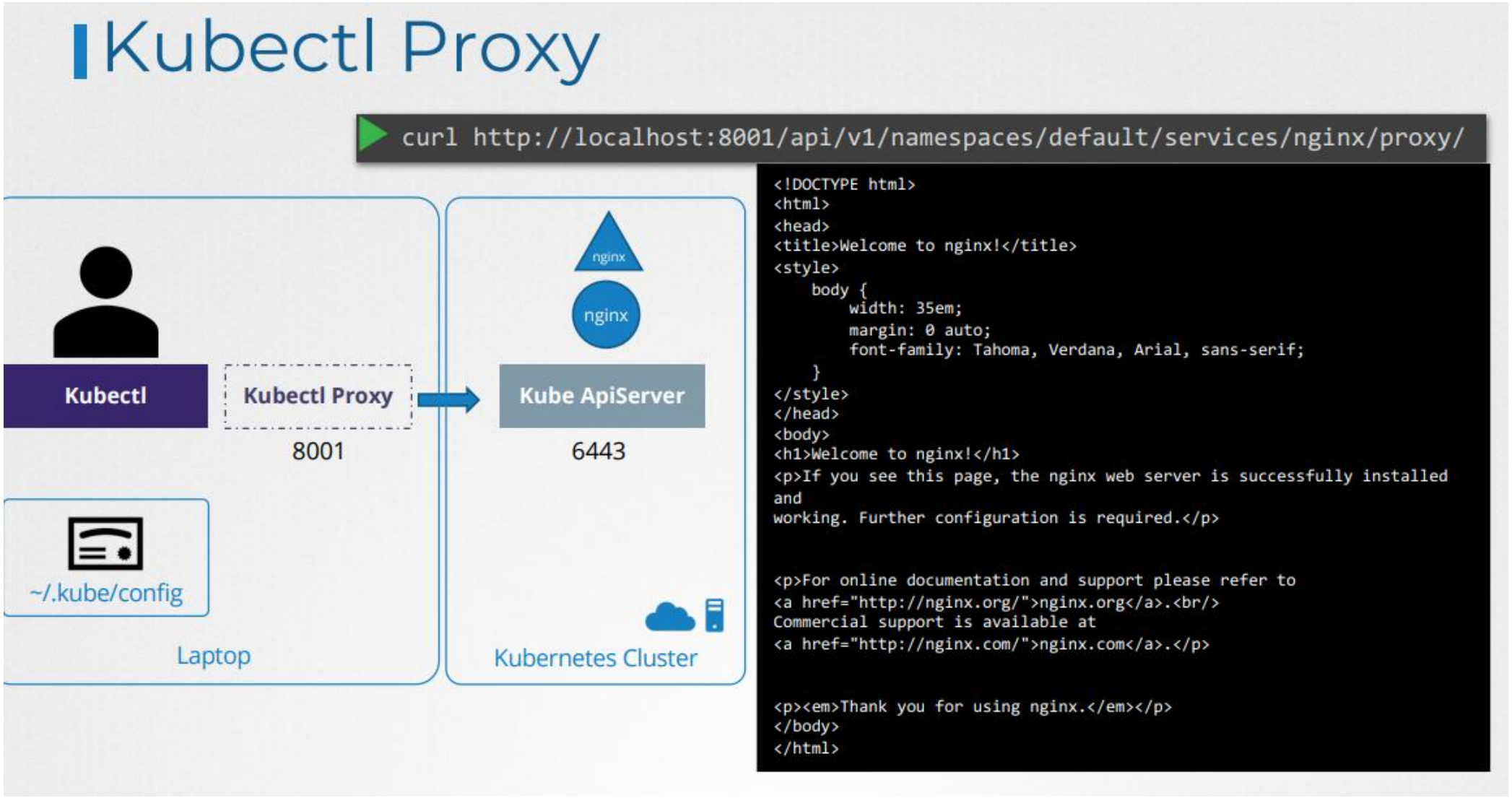
kubelet-config.yaml

```yaml
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
authentication:
  anonymous:
    enabled: false
  x509:
    clientCAFile: /path/to/ca.crt
authorization:
  mode: Webhook
readOnlyPort: 0
```

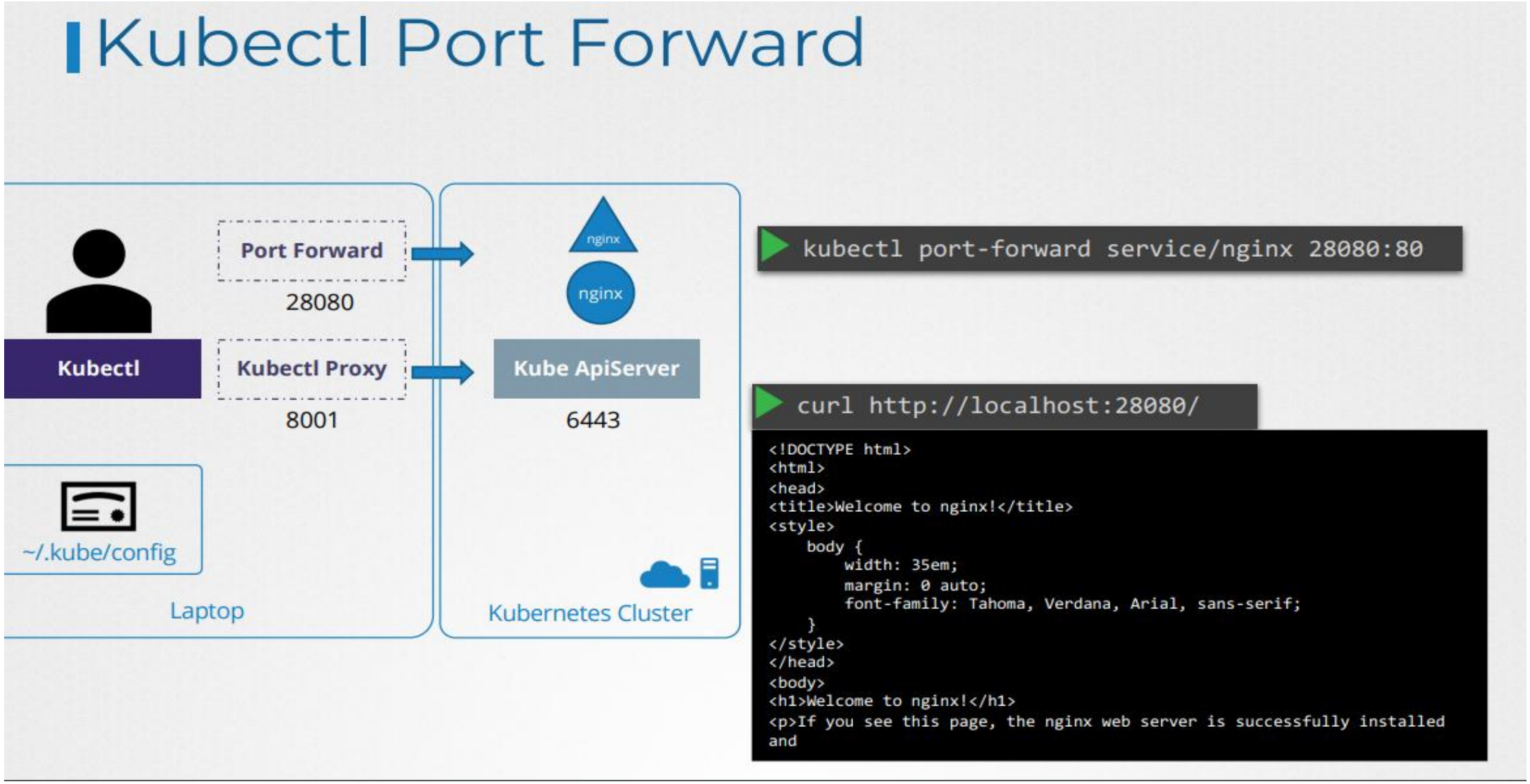
Kubectl proxy and port forwarding



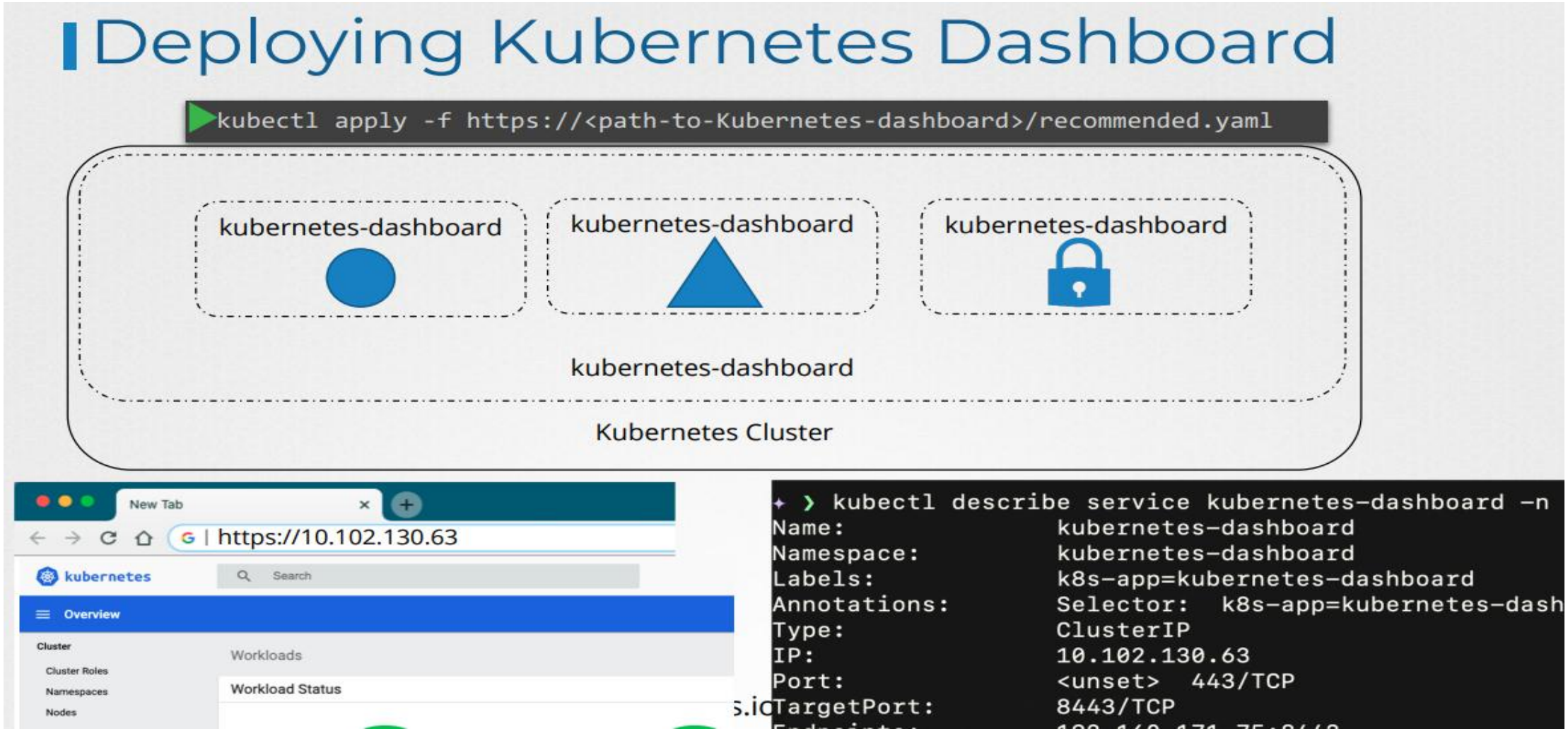
Kubectl proxy and port forwarding



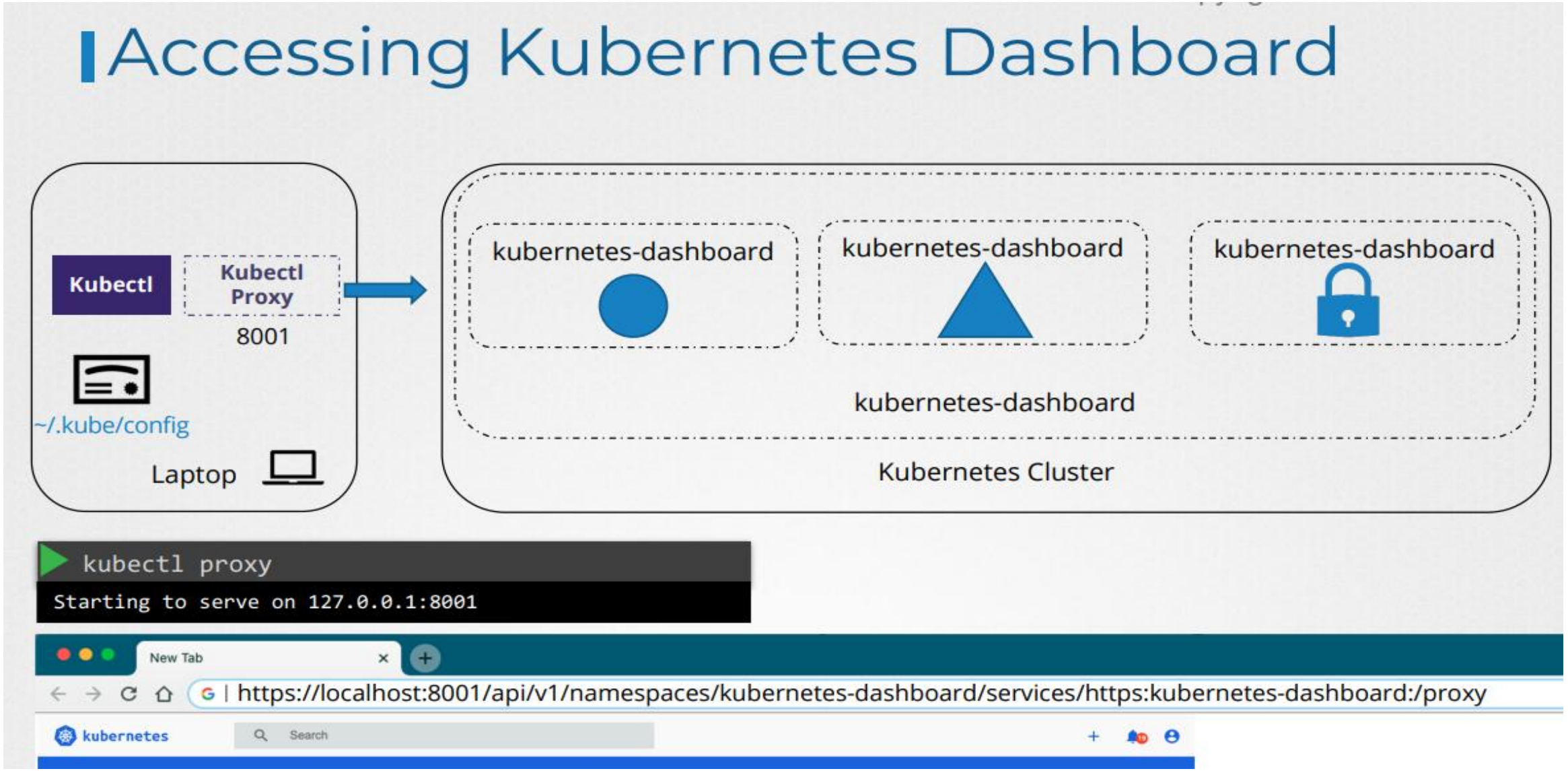
Kubectl proxy and port forwarding



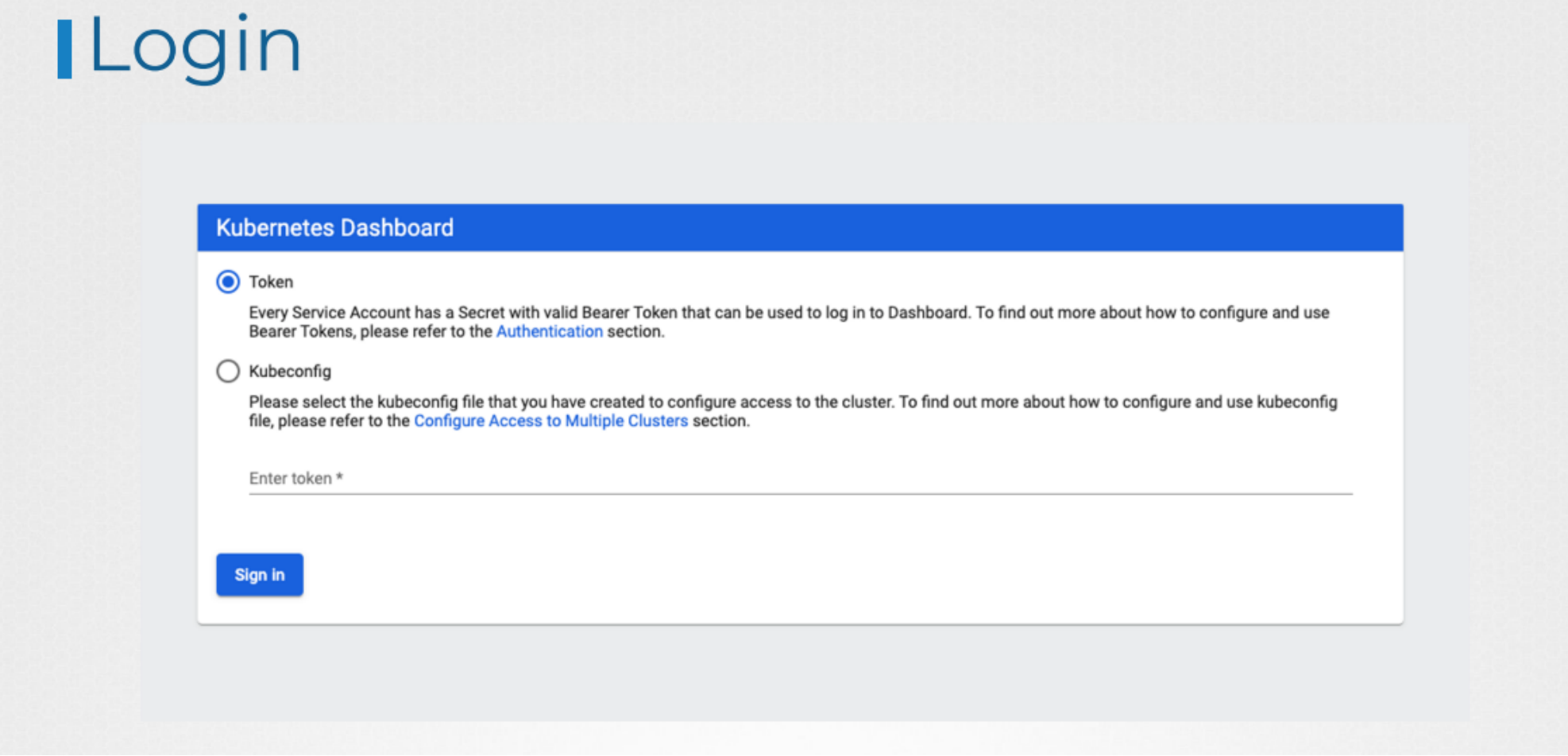
Kubernetes Dashboard Security



Kubernetes Dashboard Security



Kubernetes Dashboard Security



Kubernetes Dashboard Security

Creating sample user

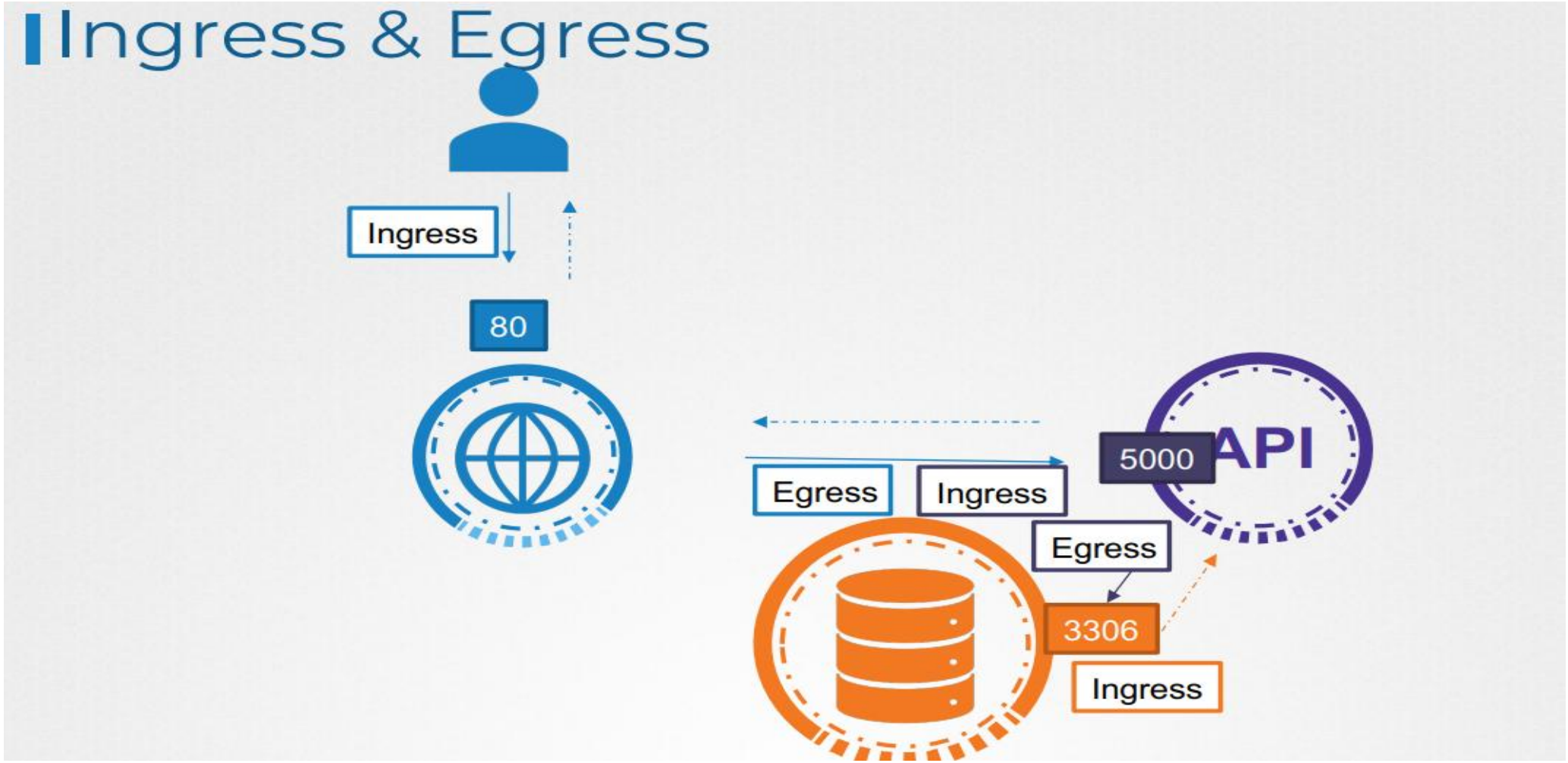
Creating a Service Account

We are creating Service Account with name `admin-user` in namespace `kubernetes-dashboard` first.

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
EOF
```

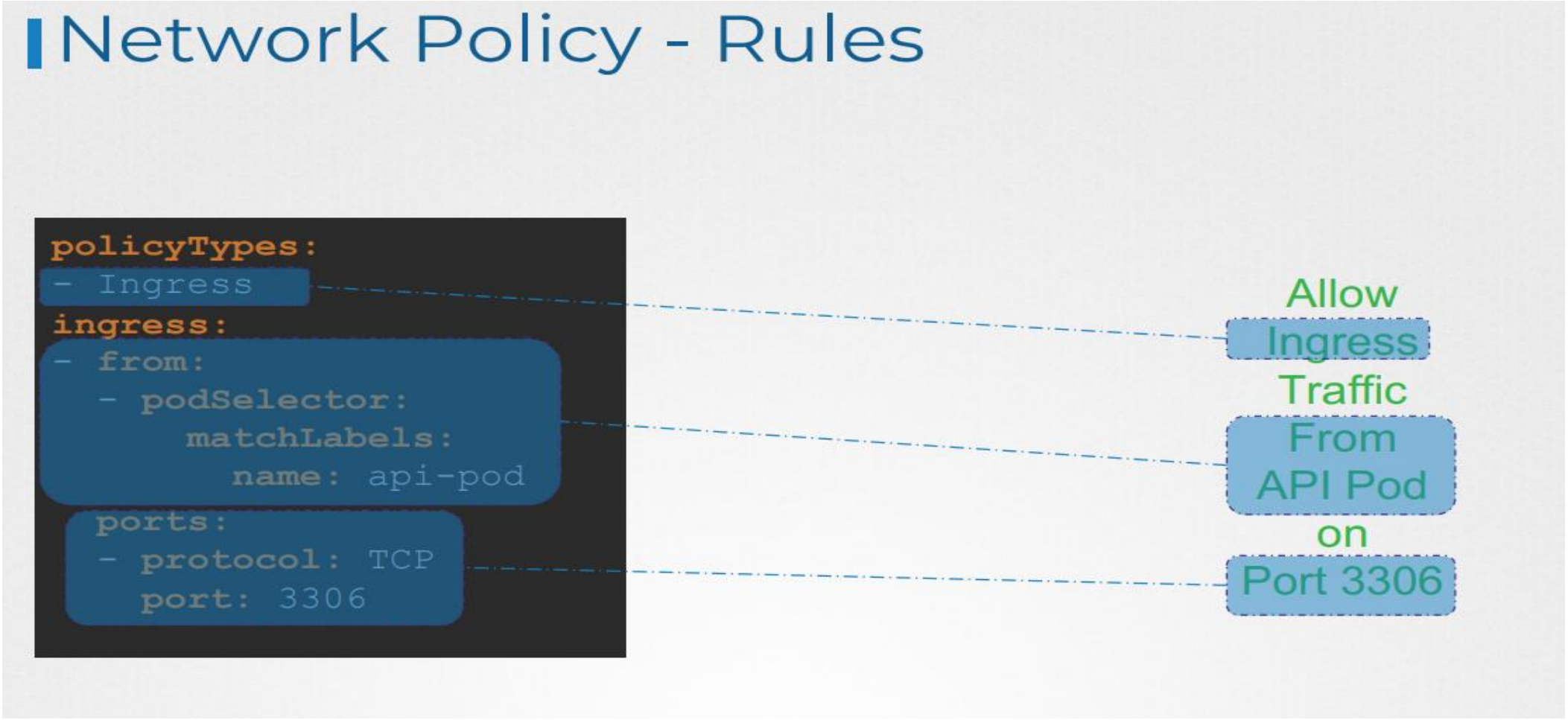
```
cat <<EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
EOF
```

Network Policy



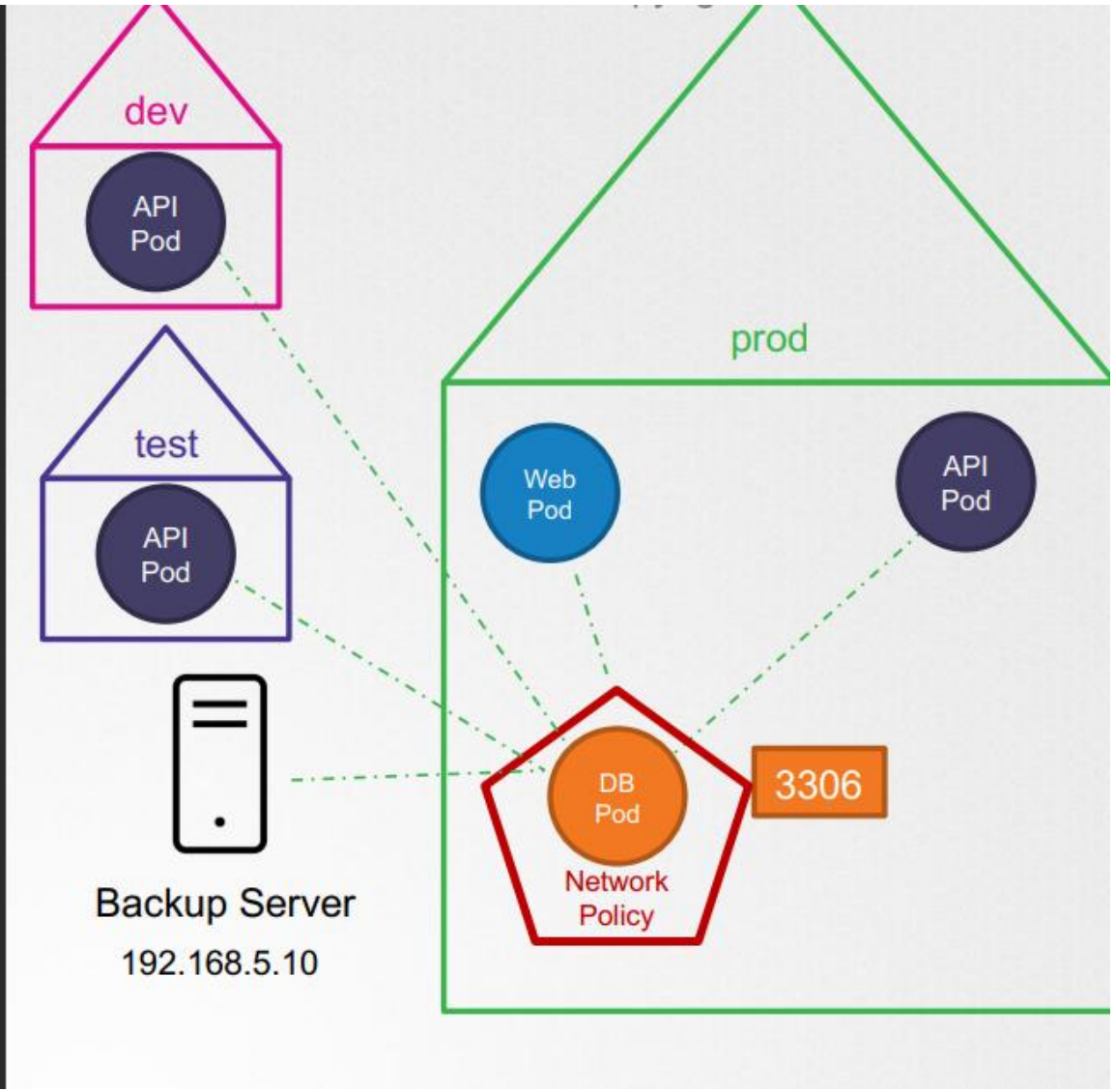
Network Policy

Network Policy

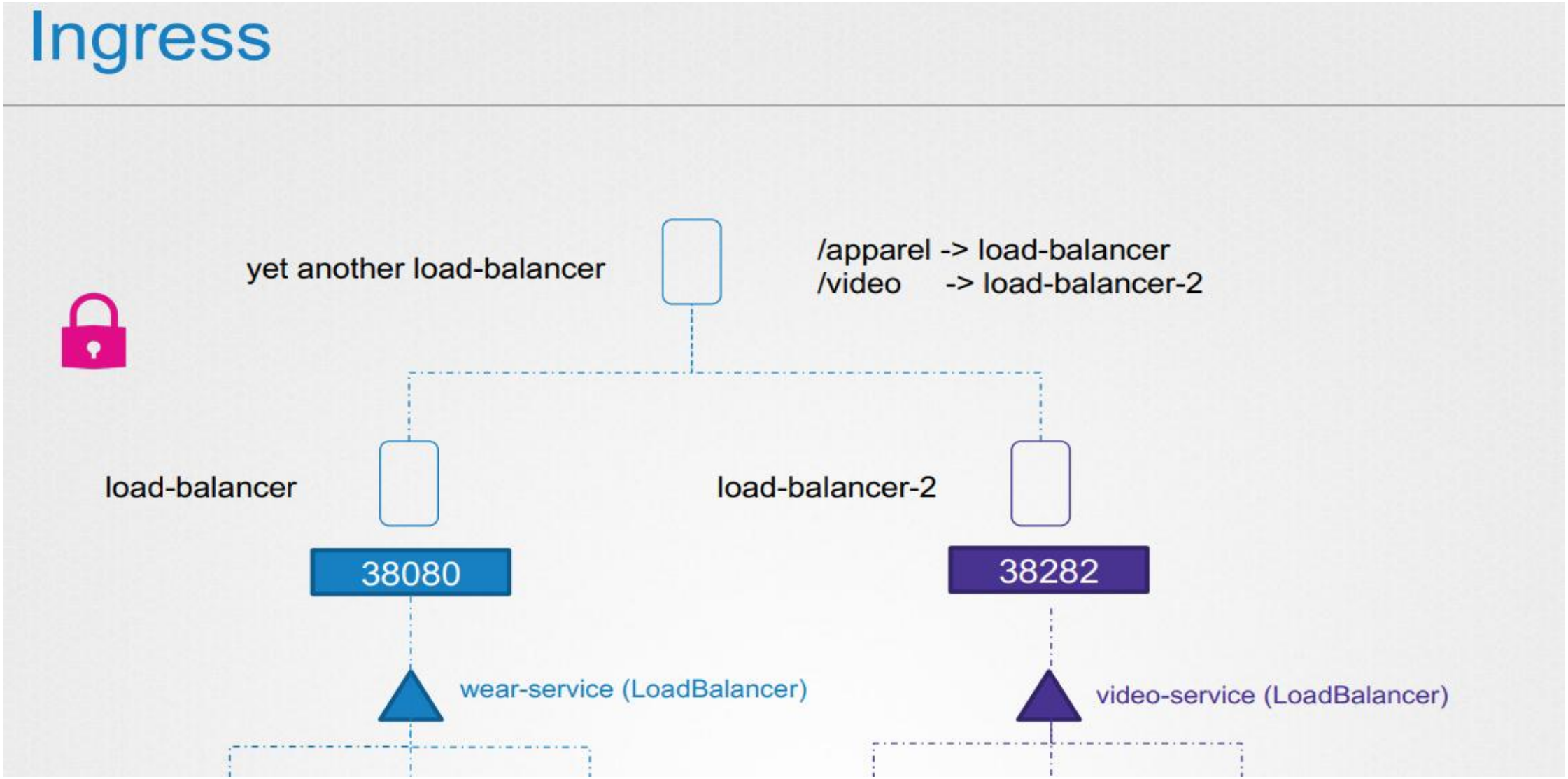


Network Policy

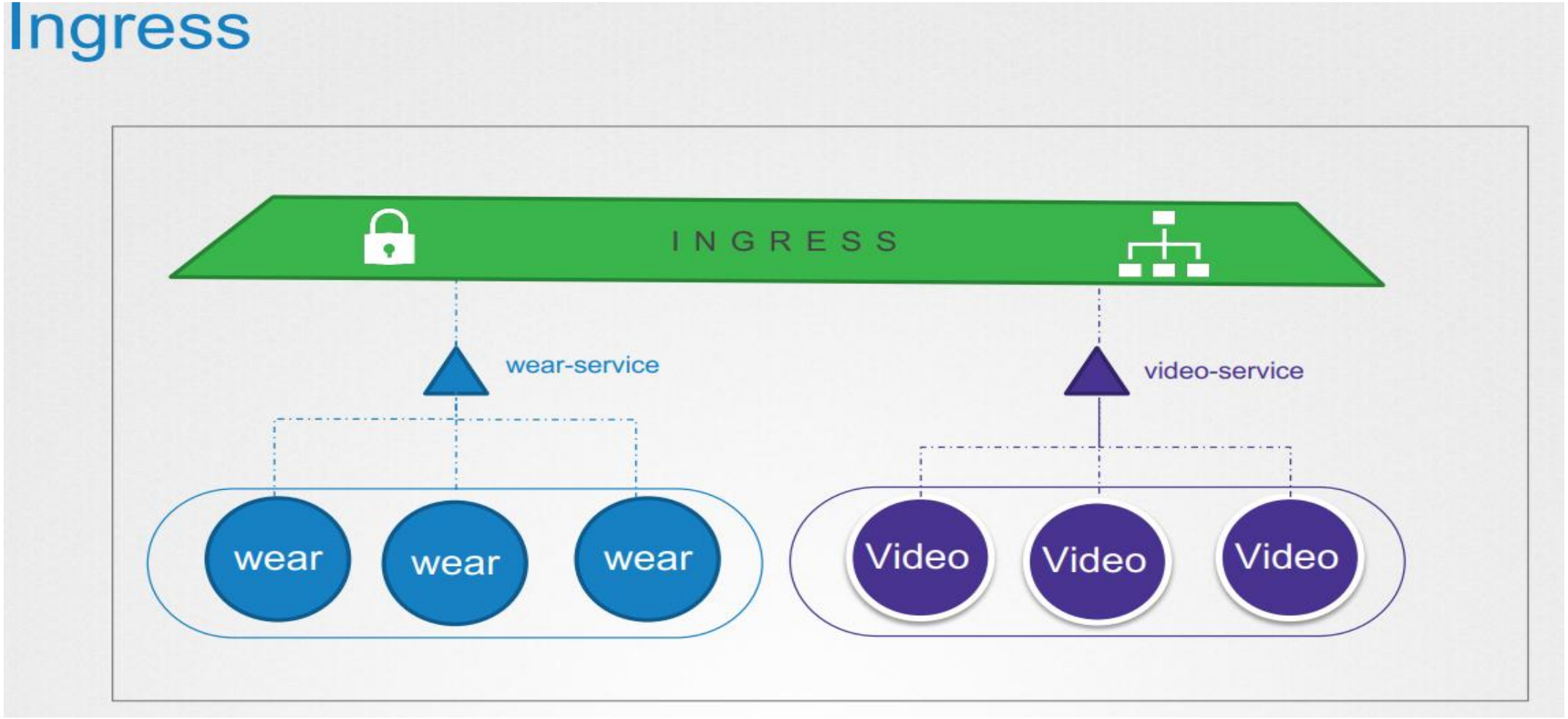
```yaml
podSelector:
  matchLabels:
    role: db
policyTypes:
- Ingress

ingress:
- from:
  - podSelector:
      matchLabels:
        name: api-pod
  - namespaceSelector:
      matchLabels:
        name: prod
  - ipBlock:
      cidr: 192.168.5.10/32


  ports:
  - protocol: TCP
    port: 3306
```



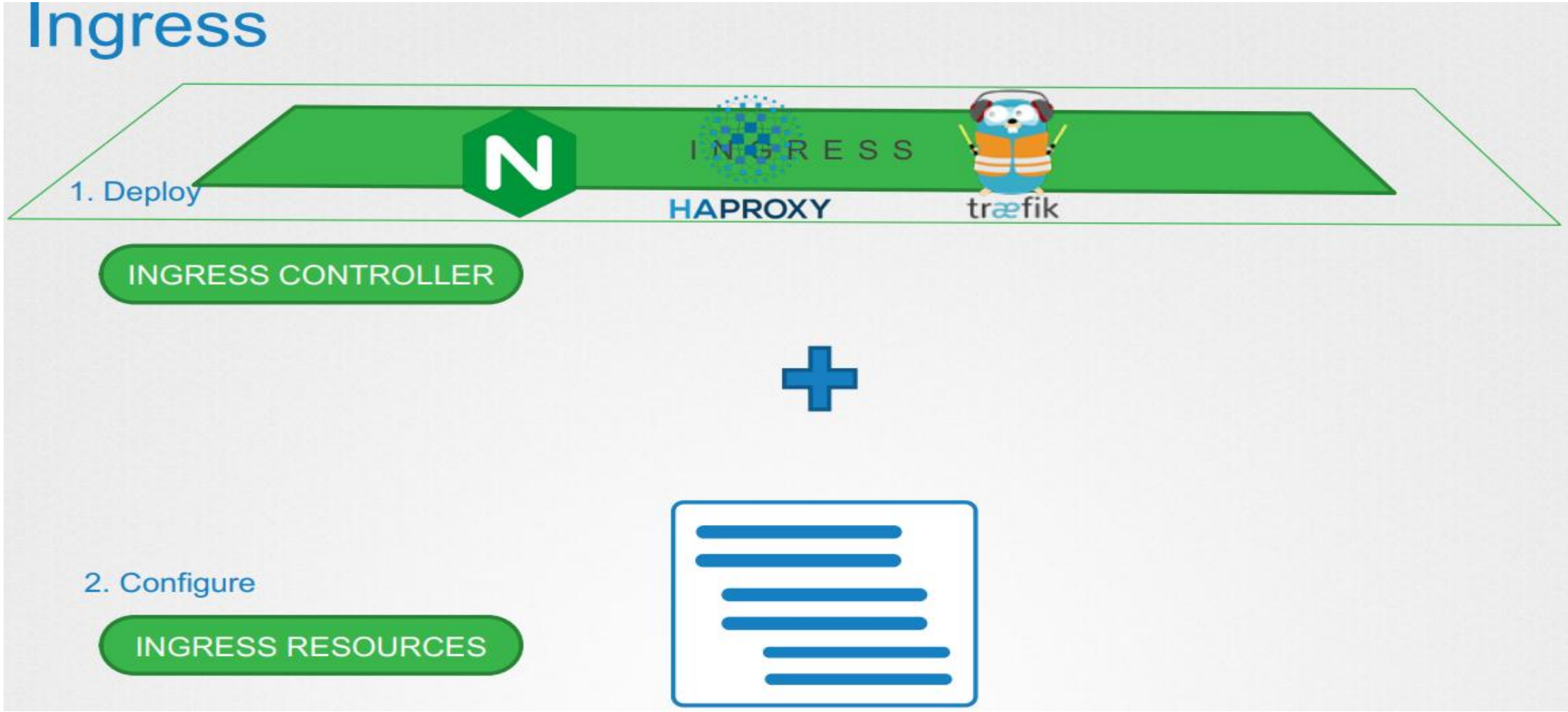
Ingress


Ingress
yet another load-balancer
/apparel -> load-balancer
/video -> load-balancer-2
load-balancer
load-balancer-2
38080
38282
wear-service (LoadBalancer)
video-service (LoadBalancer)

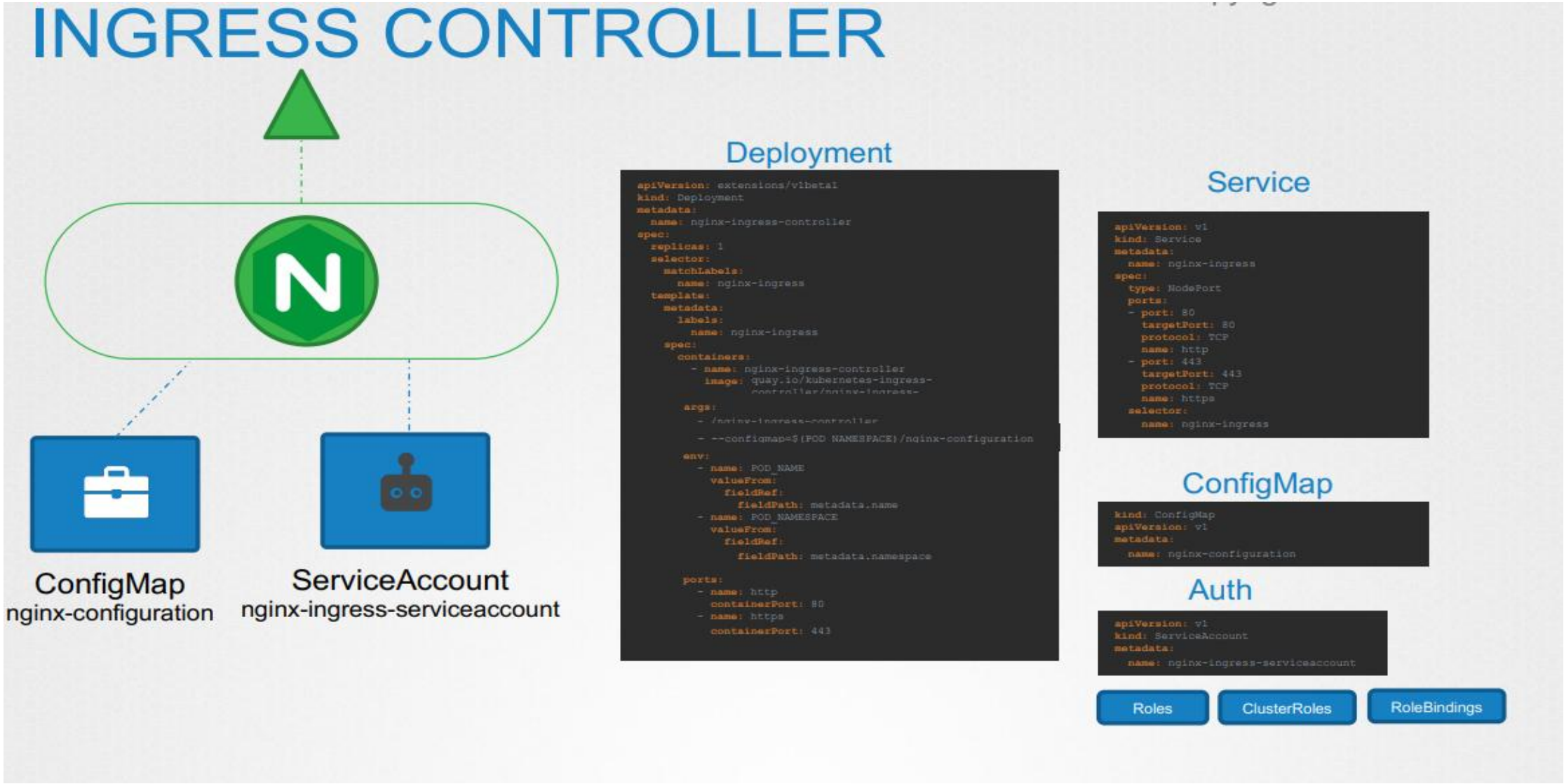
Ingress


Ingress
INGRESS
wear-service
video-service
wear
wear
wear
Video
Video
Video

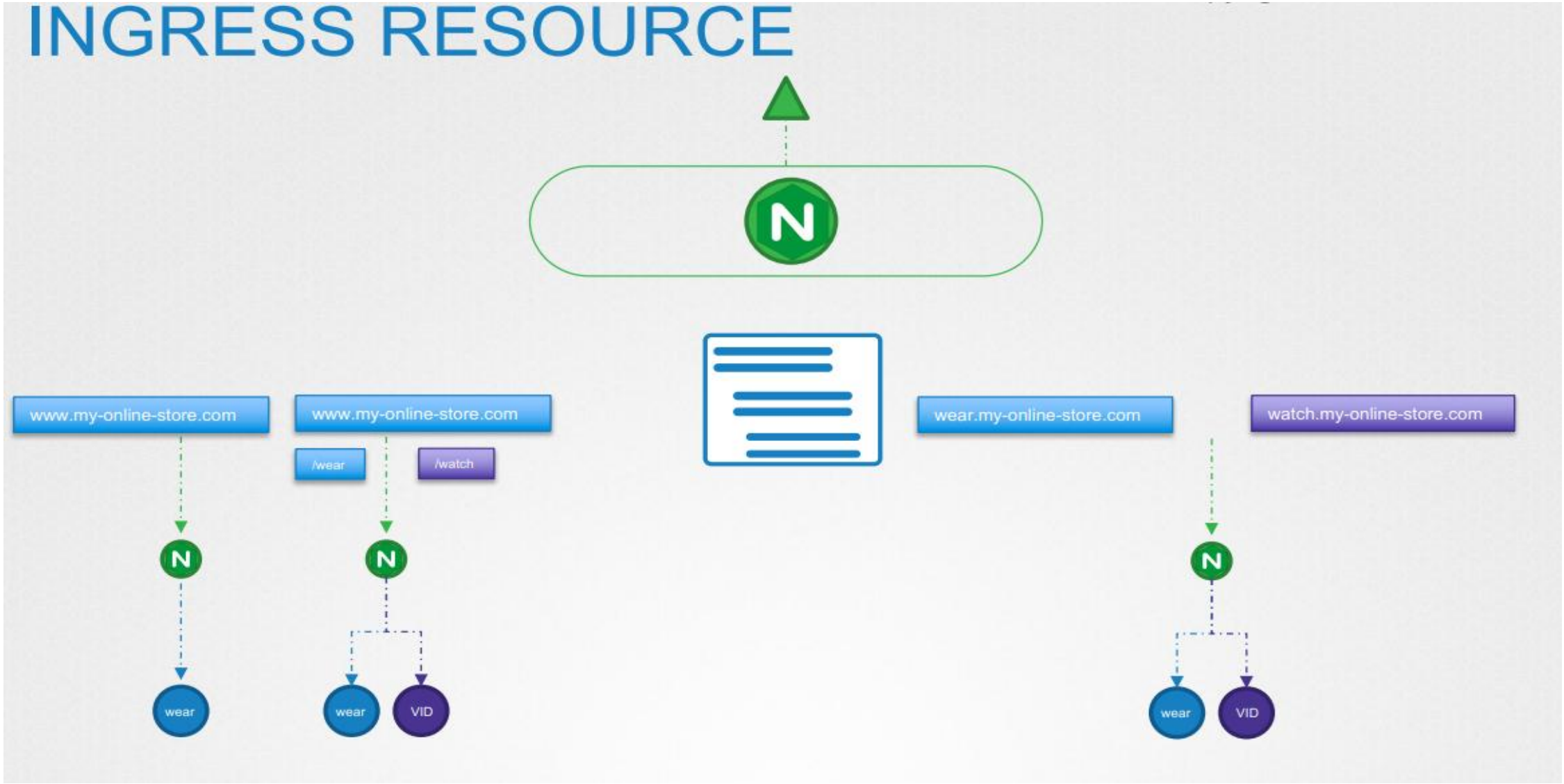
Ingress


Ingress
1. Deploy
INGRESS
HAPROXY
træfik
INGRESS CONTROLLER
2. Configure
INGRESS RESOURCES

Ingress

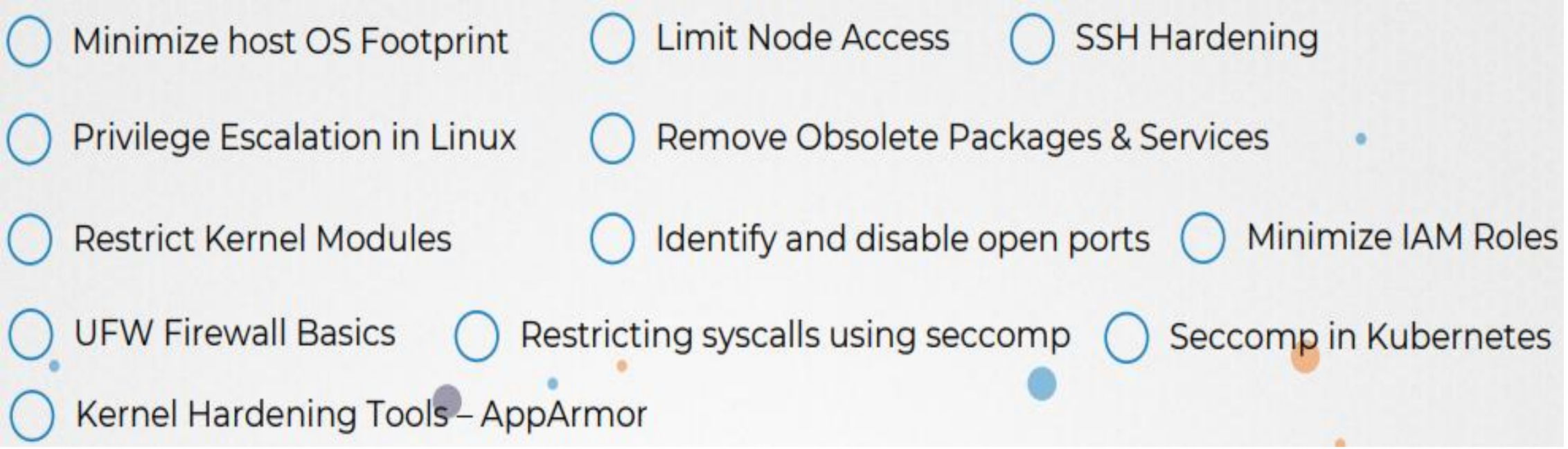

INGRESS CONTROLLER
Deployment
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
name: nginx-ingress-controller
spec:
replicas: 1
selector:
matchLabels:
name: nginx-ingress
template:
metadata:
labels:
name: nginx-ingress
spec:
containers:
- name: nginx-ingress-controller
image: quay.io/kubernetes-ingress-
controller/nginx-ingress-
args:
- /nginx-ingress-controller
- --configmap=${POD_NAMESPACE}/nginx-configuration
env:
- name: POD_NAME
valueFrom:
fieldRef:
fieldPath: metadata.name
- name: POD_NAMESPACE
valueFrom:
fieldRef:
fieldPath: metadata.namespace
ports:
- name: http
containerPort: 80
- name: https
containerPort: 443
Service
apiVersion: v1
kind: Service
metadata:
name: nginx-ingress
spec:
type: NodePort
ports:
- port: 80
targetPort: 80
protocol: TCP
name: http
- port: 443
targetPort: 443
protocol: TCP
name: https
selector:
name: nginx-ingress
ConfigMap
kind: ConfigMap
apiVersion: v1
metadata:
name: nginx-configuration
Auth
apiVersion: v1
kind: ServiceAccount
metadata:
name: nginx-ingress-serviceaccount
Roles
ClusterRoles
RoleBindings
ConfigMap
nginx-configuration
ServiceAccount
nginx-ingress-serviceaccount

Ingress

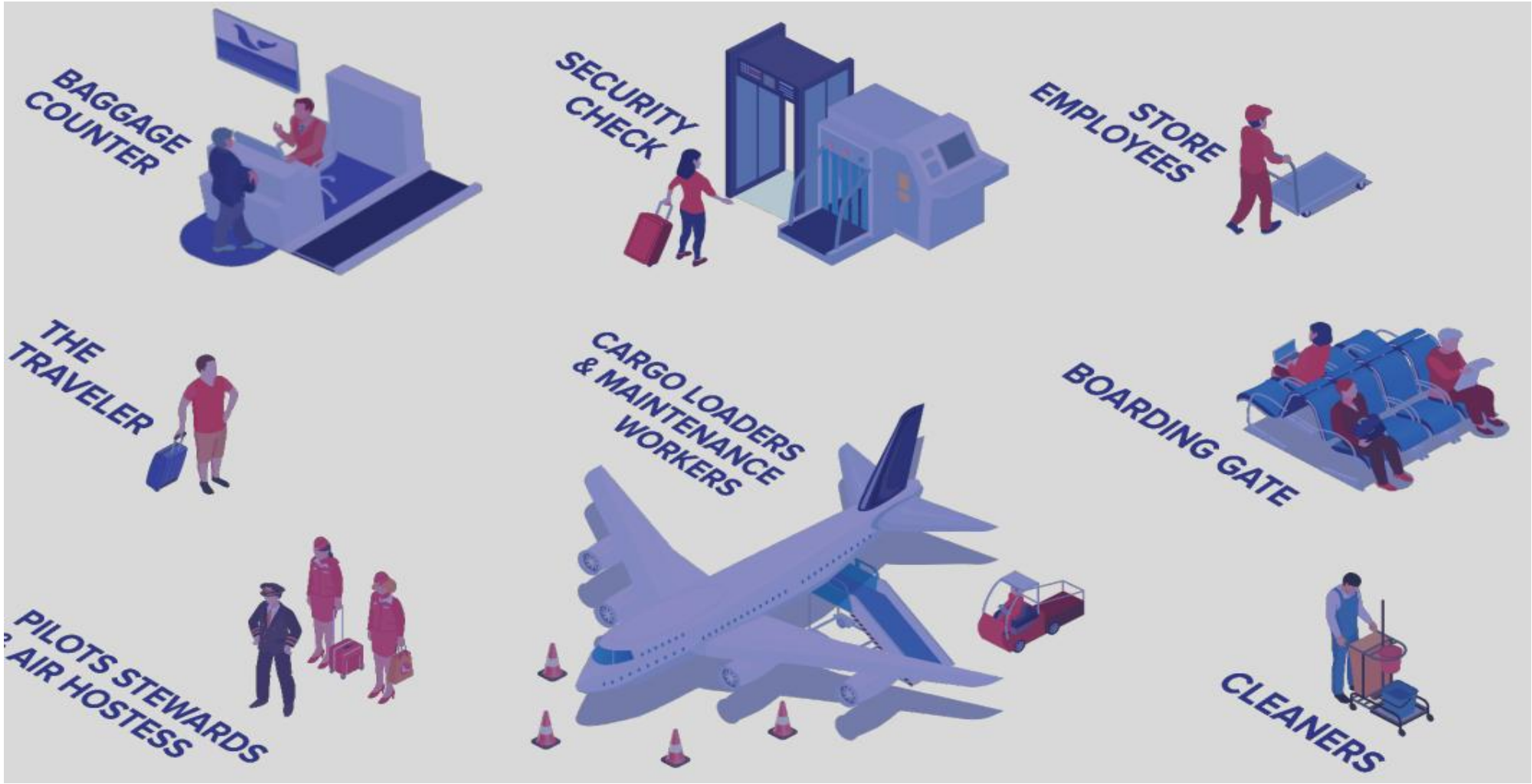

INGRESS RESOURCE
www.my-online-store.com
www.my-online-store.com
/wear
/watch
wear.my-online-store.com
watch.my-online-store.com
wear
wear
VID
wear
VID

System Hardening

- System Hardening

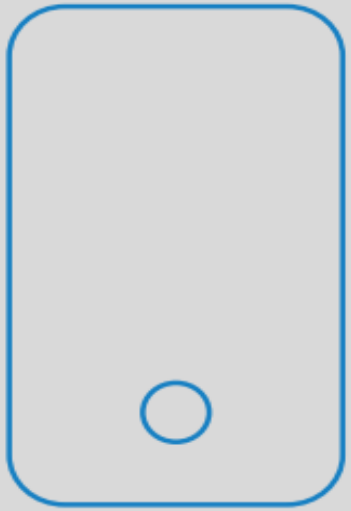


Least privileges principle

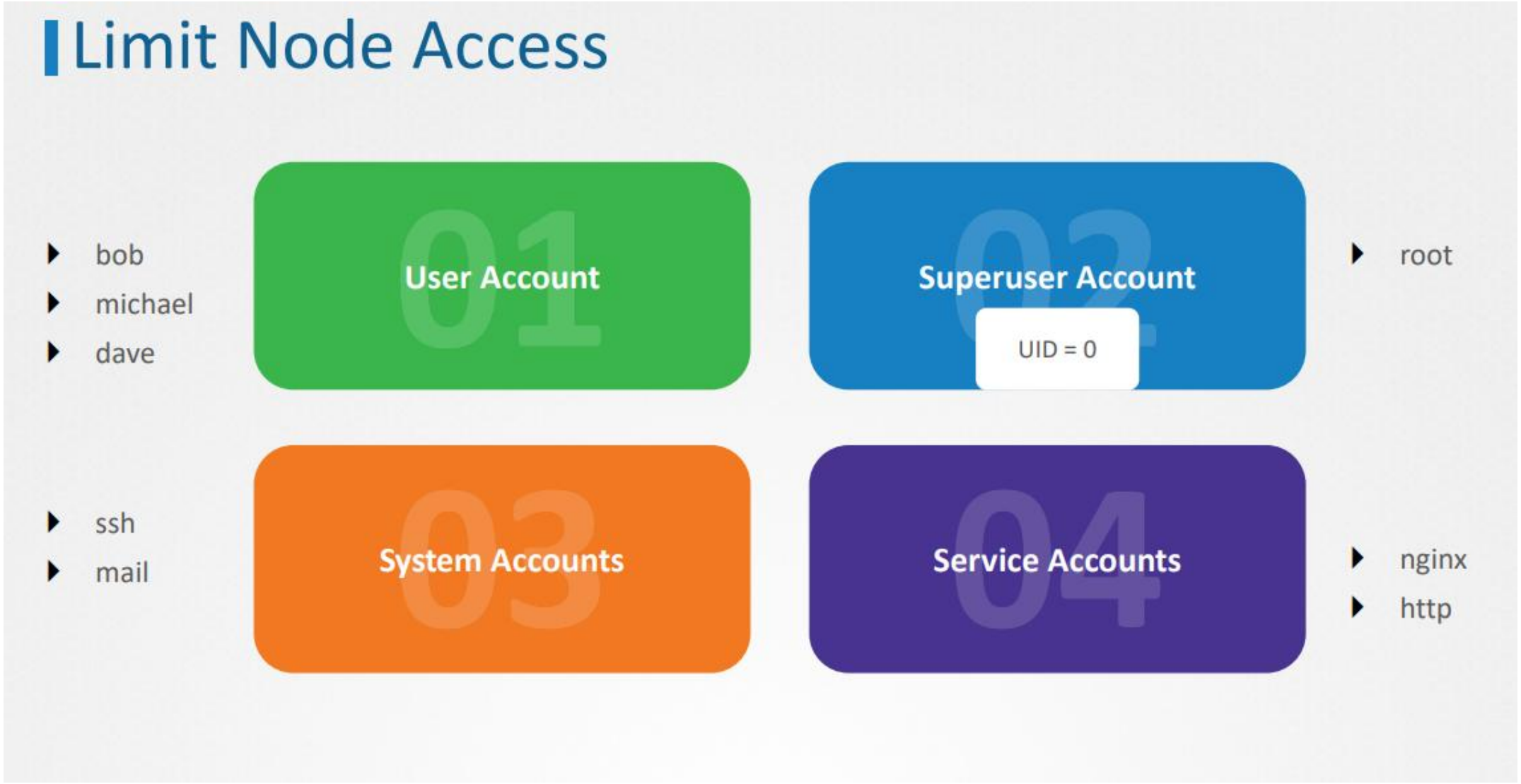


Least privileges principle

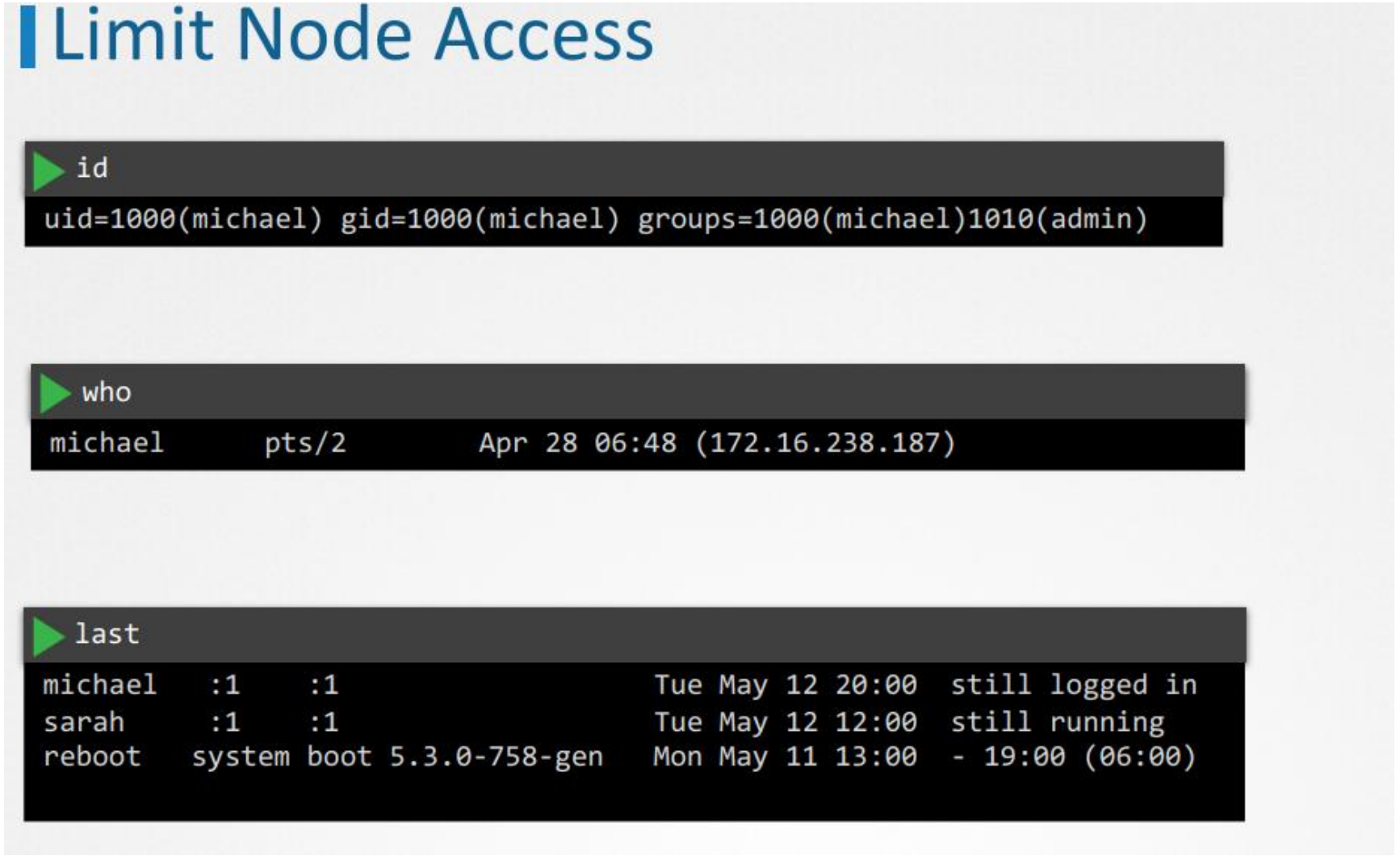
- Limit Access to Nodes
- RBAC Access
- Remove Obsolete Packages & Services
- Restrict Network Access
- Restrict Obsolete Kernel Modules
- Identify and Fix Open Ports



Limit Node Access



Limit Node Access



Limit Node Access

Limit Node Access

Limit Node Access

SSH Hardening

SSH

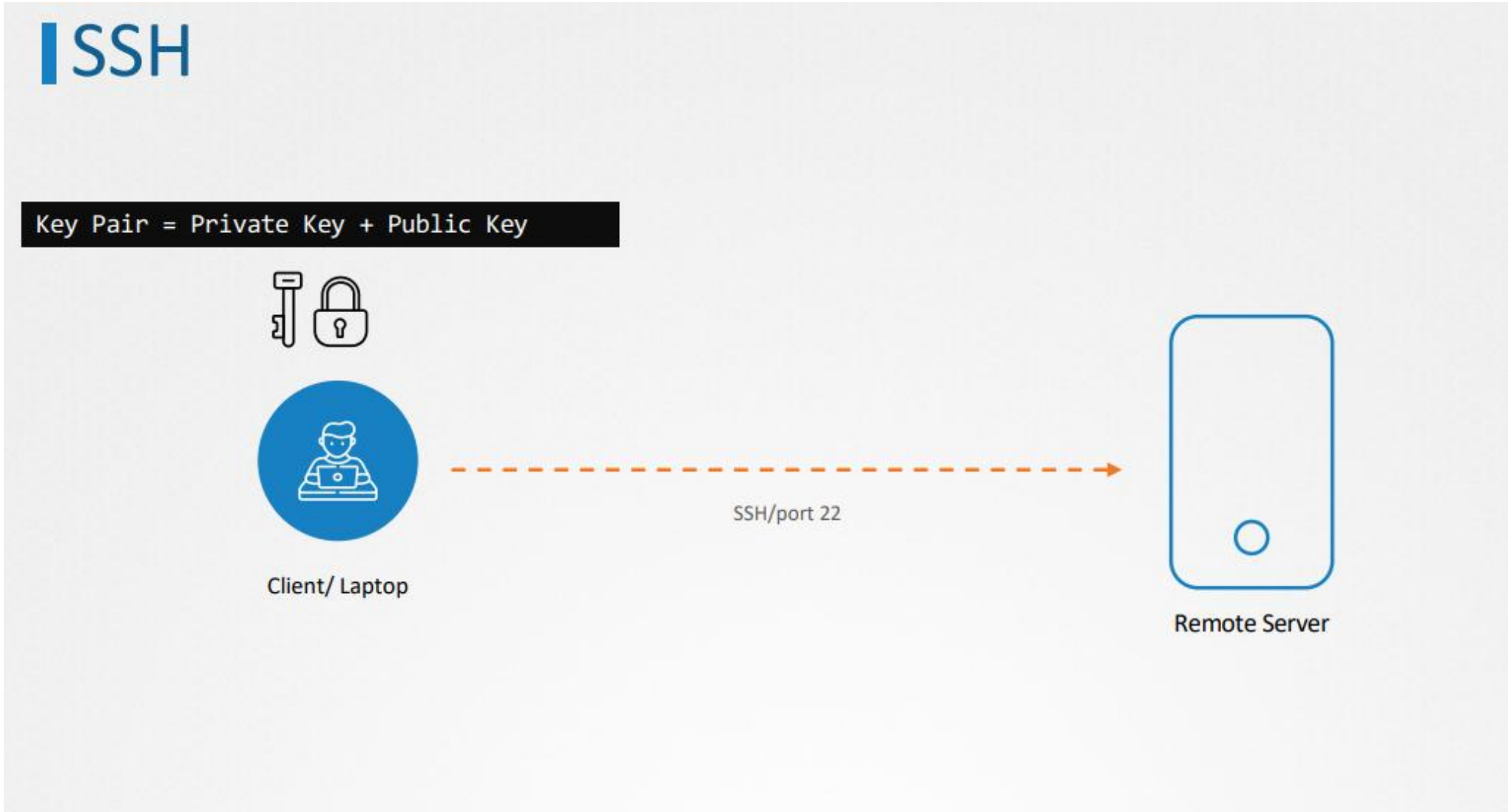
```
ssh <hostname OR IP Address>
```

```
ssh <user>@<hostname OR IP Address>
```

```
ssh -l <user> <hostname OR IP Address>
```


```
[mark@localhost ~]$ ssh node01
mark@node01's password:
Last login: Tue Apr  7 20:08:58 2020 from 192.168.1.109
[mark@node01 ~]$
```

SSH Hardening



SSH Hardening

SSH

Client/ Laptop

Public Key: /home/mark/.ssh/id_rsa.pub

Private Key: /home/mark/.ssh/id_rsa

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ mark /.ssh/id_rsa):
/home/mark/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mark/.ssh/id_rsa.
Your public key has been saved in /home/mark/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:PCRTdbxxzffzmi8uunjn5V/1LZCG0BvhVJYXBr9gYsE mark@localhost
The key's randomart image is:
+---[RSA 2048]----+
|         .o=o=oo+ |
|        . +E=+oo +|
|       o o * o=. o|
|        = o *.o o.|
|         S o + . +|
|          . . .  =|
|               oo+|
|          .. oo+..|
|         ..o=.oo+o|
+----[SHA256]-----+
```

SSH Hardening

SSH

Client/ Laptop

```
ssh-copy-id mark@node01
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/mark/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new
key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed --
if you are prompted now it is to install the new keys
mark@node01's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'mark@node01'"
and check to make sure that only the key(s) you wanted were
added.
```

```
ssh node01
Last login: Tue Apr  7 20:10:58 2020 from 192.168.1.109
[mark@node01 ~]$
```

SSH Hardening

SSH

```
cat /home/mark/.ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCgVV5wgH37kNwjnEIxgeX4j6LASNckjKi4bRpjPGecyxEi
EeJhIU4x31XPEFzUFp/1xX2rjeiM2Ko3oPmTGCCTEQMpQogerR7NS+bA9eXs34jWIg+xoSQjeQu1
+lXgrRippJn2YhWYVAY3sKWIiiklowuMXmxjmBBr48L52di1J+8EASwnM4ILX/YL72Czq3uFFhVW
1fNUKBPUbW58h4QSAd2r9abzZfrHH48ThPJW4/5i8LOHEo3W0BXl3foEV0c6pk3TgxcjTuZQOimd
48mM2pxWJh9WxA0xcXwbD3+JrcnZeMJq4TbrKjaXQ0pBGenglxurxnRT2og9DeTIqGN3
mark@localhost
```

Remote Server

SSH Hardening

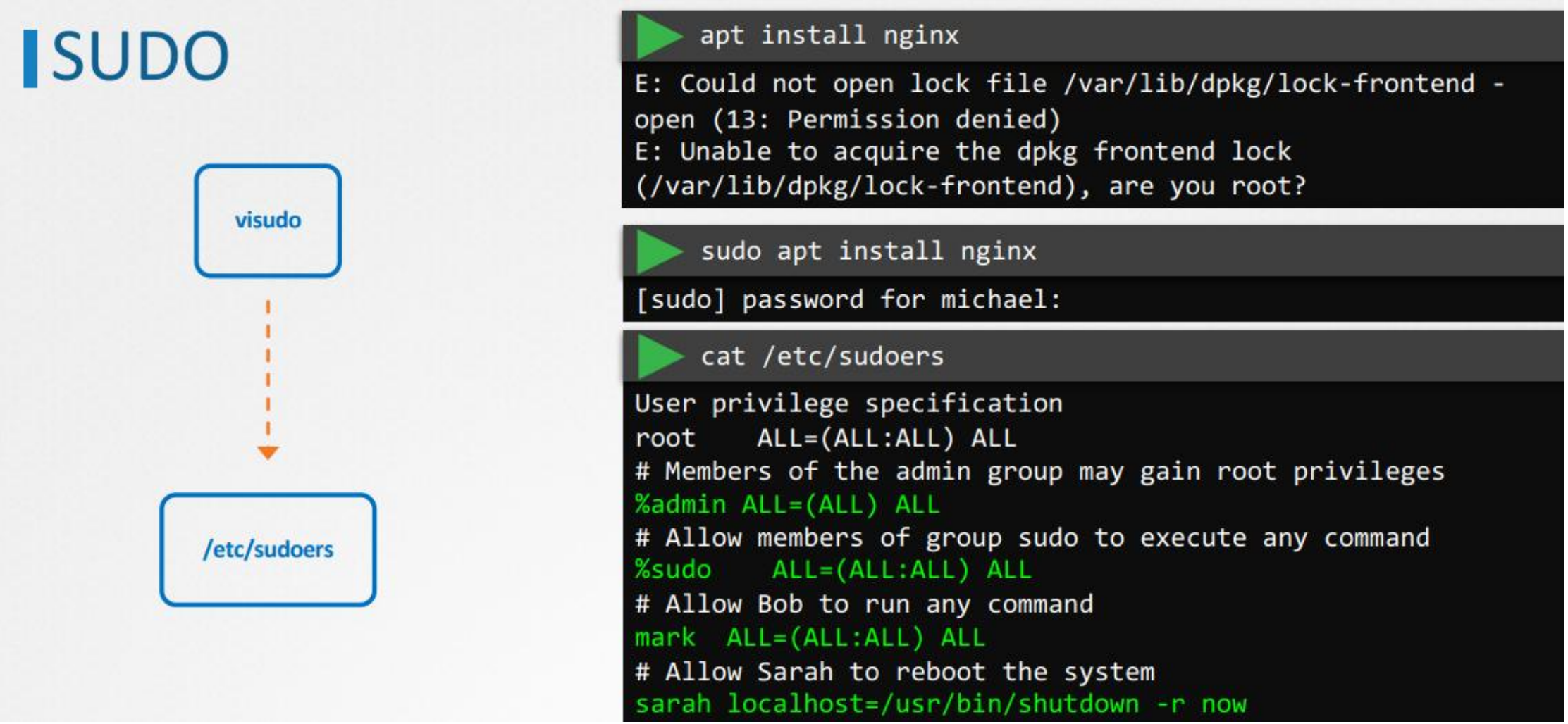
HARDEN SSH SERVICE

```
vi /etc/ssh/sshd_config
PermitRootLogin no
PasswordAuthentication no
```

```
systemctl restart sshd
```

Remote Server

Privilege Escalation



Privilege Escalation

SUDO

```
cat /etc/sudoers
User privilege specification
root    ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
# Allow members of group sudo to execute any command
%sudo    ALL=(ALL:ALL) ALL
# Allow Bob to run any command
mark  ALL=(ALL:ALL) ALL
# Allow Sarah to reboot the system
sarah localhost=/usr/bin/shutdown -r now
# See sudoers(5) for more information on "#include"
directives:
#includedir /etc/sudoers.d
```

Field	Description	Example
1	User or Group	bob, %sudo (group)
2	Hosts	localhost, ALL(default)
3	User	ALL(default)
4	Command	/bin/ls, ALL(unrestricted)

Remove obsolete packages

Remove obsolete packages

Remove Unwanted Services

```
systemctl list-units --type service
```

```
Apache2.service                    loaded active running The Apache HTTP Server
apparmor.service                   loaded active exited  AppArmor initialization
containerd.service                 loaded active running containerd container runtime
dbus.service                       loaded active running D-Bus System Message Bus
docker.service                     loaded active running Docker Application Container Engine
ebtables.service                   loaded active exited  ebtables ruleset management
kmod-static-nodes.service          loaded active exited  Create list of required static device n
kubelet.service                    loaded active running kubelet: The Kubernetes Node Agent
proxy.service                      loaded active running kubectl proxy 8888
systemd-journal-flush.service      loaded active exited  Flush Journal to Persistent Storage
```

```
systemctl stop apache2
```

```
systemctl disable apache2
```

```
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysv-
install.
Executing: /lib/systemd/systemd-sysv-install disable apache2
```

Remove obsolete packages

CIS Benchmark Reference

2 Services

While applying system updates and patches helps correct known vulnerabilities, one of the best ways to protect the system against as yet unreported vulnerabilities is to disable all services that are not required for normal system operation. This prevents the exploitation of vulnerabilities discovered at a later date. If a service is not enabled, it cannot be exploited. The actions in this section of the document provide guidance on some services which can be safely disabled and under which circumstances, greatly reducing the number of possible threats to the resulting system. Additionally some services which should remain enabled but with secure configuration are covered as well as insecure service clients.

Identify and Disable open ports

Disable Open Ports


```
systemctl status ssh
```

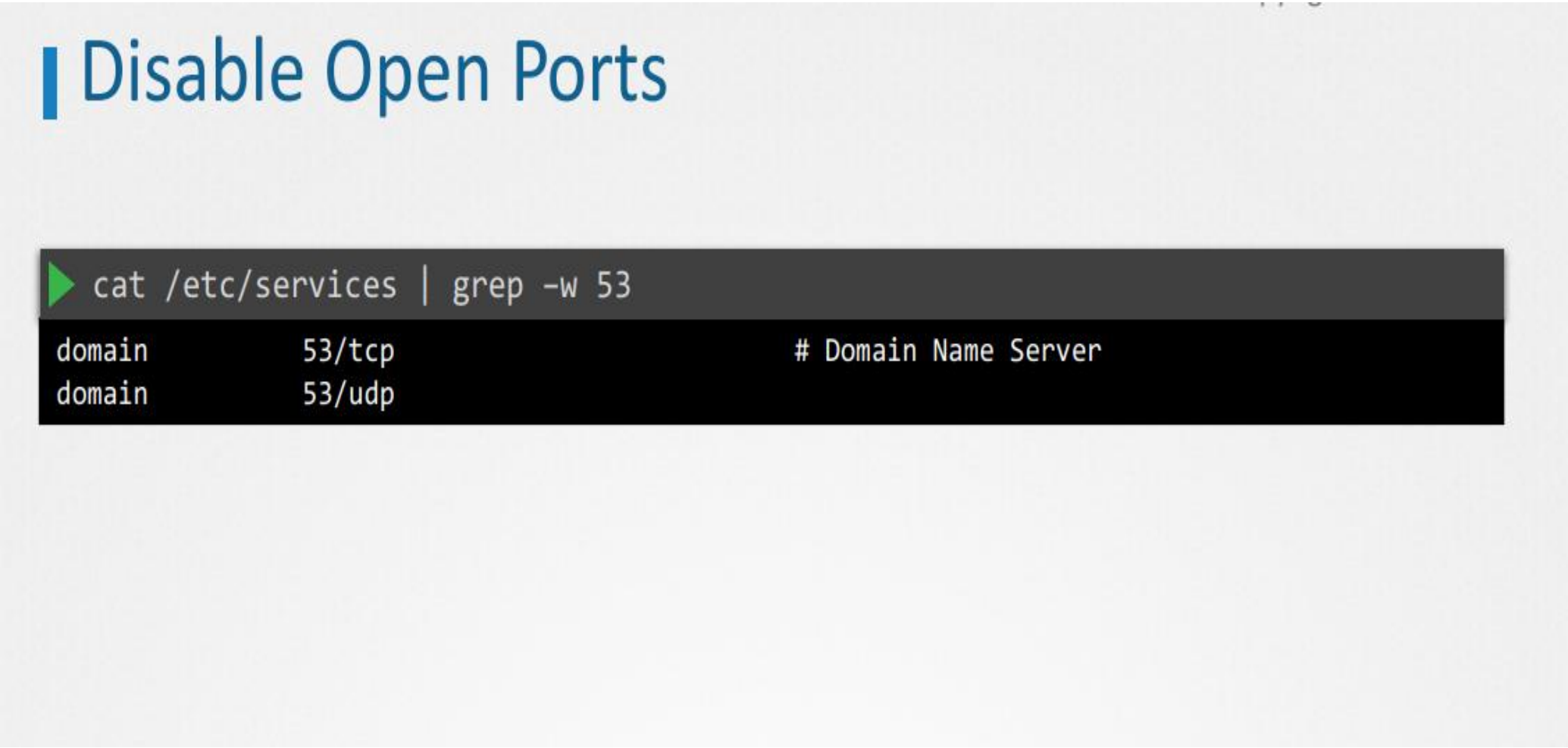
```
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-03-17 08:33:29 UTC; 54min ago
 Main PID: 759 (sshd)
    Tasks: 1 (limit: 7372)
   CGroup: /system.slice/ssh.service
           └─759 /usr/sbin/sshd -D
```

Identify and Disable open ports

Disable Open Ports

```
netstat -an | grep -w LISTEN
tcp        0      0 127.0.0.1:10248         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:10249         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:2379          0.0.0.0:*               LISTEN
tcp        0      0 10.53.64.6:2379         0.0.0.0:*               LISTEN
tcp        0      0 10.53.64.6:2380         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:42893         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:2381          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:46607        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:10257         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:10259         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp6       0      0 :::10250                :::*                    LISTEN
tcp6       0      0 :::6443                 :::*                    LISTEN
tcp6       0      0 :::10256                :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
```

Identify and Disable open ports



Identify and Disable open ports

Disable Open Ports

Control-plane node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services†	All

† Default port range for NodePort Services.

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/#check-required-ports

Minimize external access to Network

Restrict Network Access

```
systemctl status ssh
```

```
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-03-17 08:33:29 UTC; 54min ago
 Main PID: 759 (sshd)
    Tasks: 1 (limit: 7372)
   CGroup: /system.slice/ssh.service
           └─759 /usr/sbin/sshd -D
```

```
cat /etc/services| grep ssh
```

```
ssh             22/tcp                          # SSH Remote Login Protocol
```

```
netstat -an | grep 22 | grep -w LISTEN
```

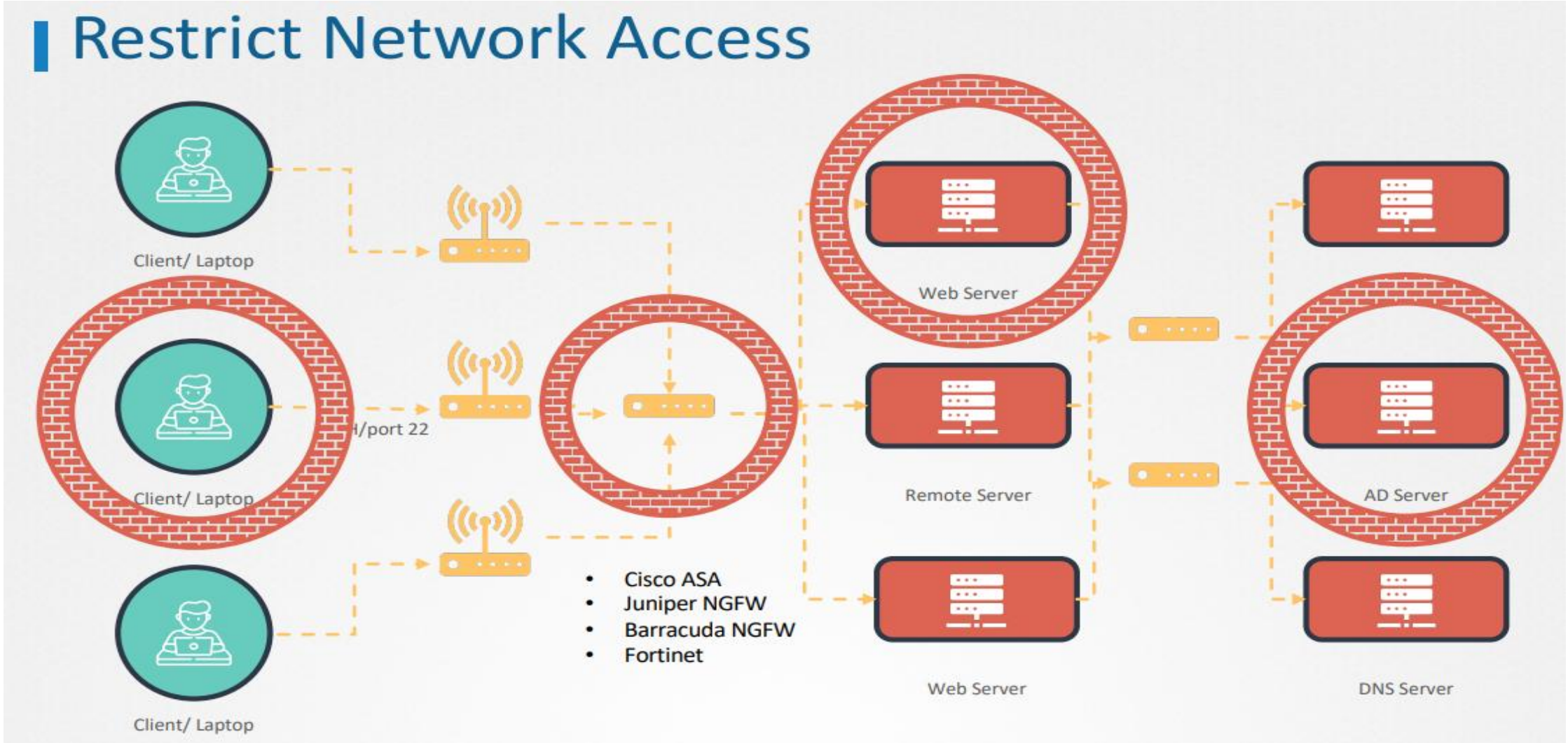
```
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
```

Minimize external access to Network

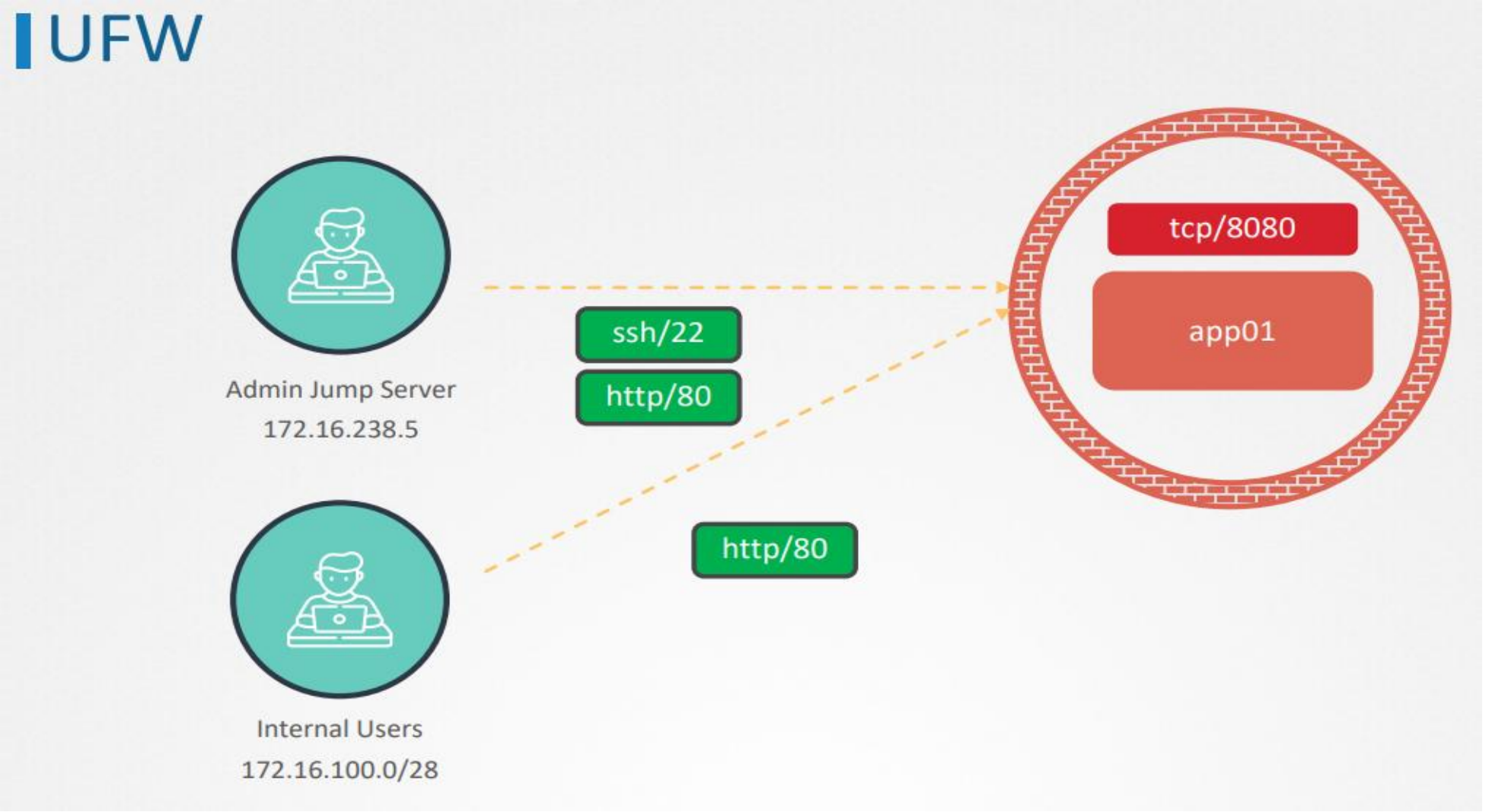
Restrict Network Access

```
netstat -an | grep -w LISTEN
tcp        0      0 127.0.0.1:10248         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:10249         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:2379          0.0.0.0:*               LISTEN
tcp        0      0 10.53.64.6:2379         0.0.0.0:*               LISTEN
tcp        0      0 10.53.64.6:2380         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:42893         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:2381          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:46607        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:10257         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:10259         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp6       0      0 :::10250                :::*                    LISTEN
tcp6       0      0 :::6443                 :::*                    LISTEN
tcp6       0      0 :::10256                :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::8888                 :::*                    LISTEN
```

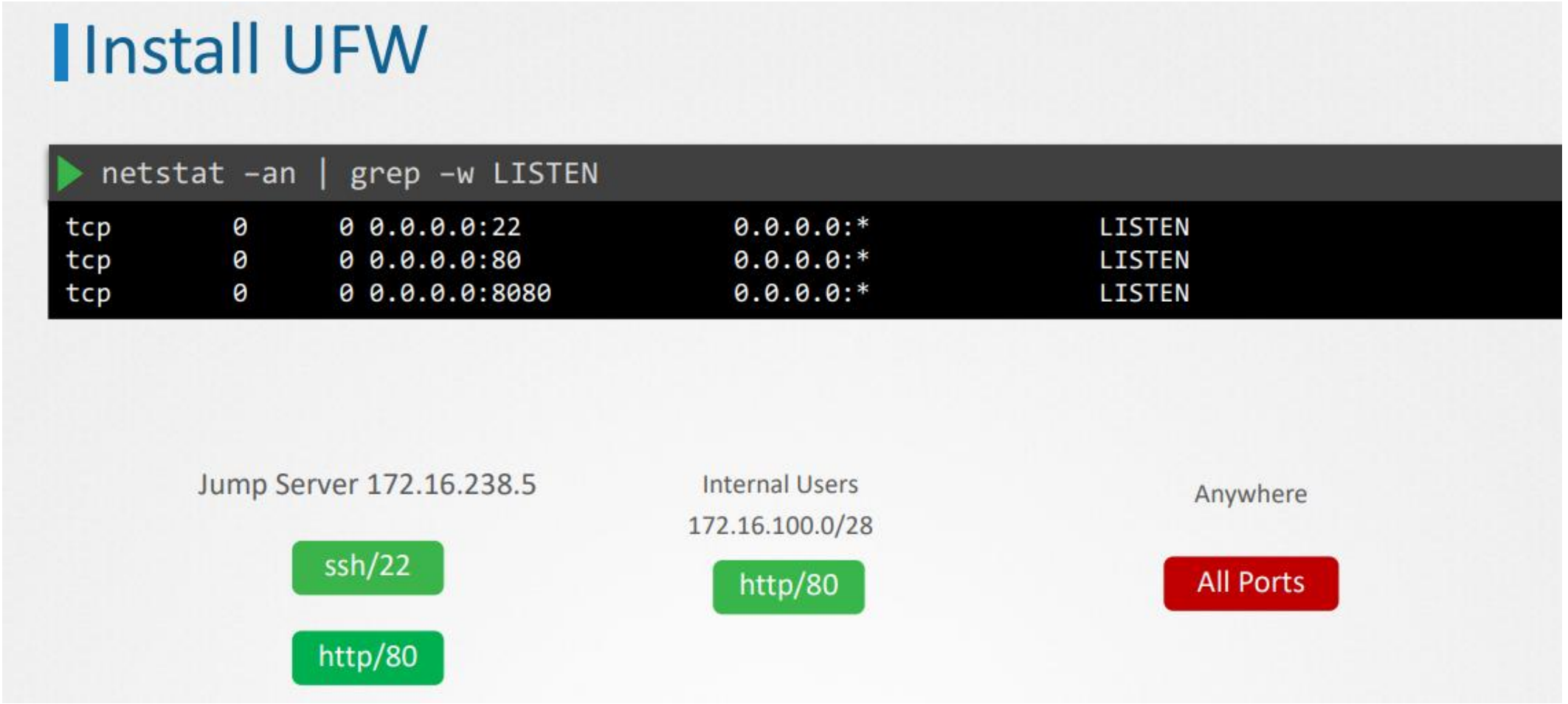
Minimize external access to Network



UFW Firewall basics



UFW Firewall basics



UFW Firewall basics

Install UFW

```
apt-get update
```

```
.
.
Fetched 22.3 MB in 6s (3449 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
21 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

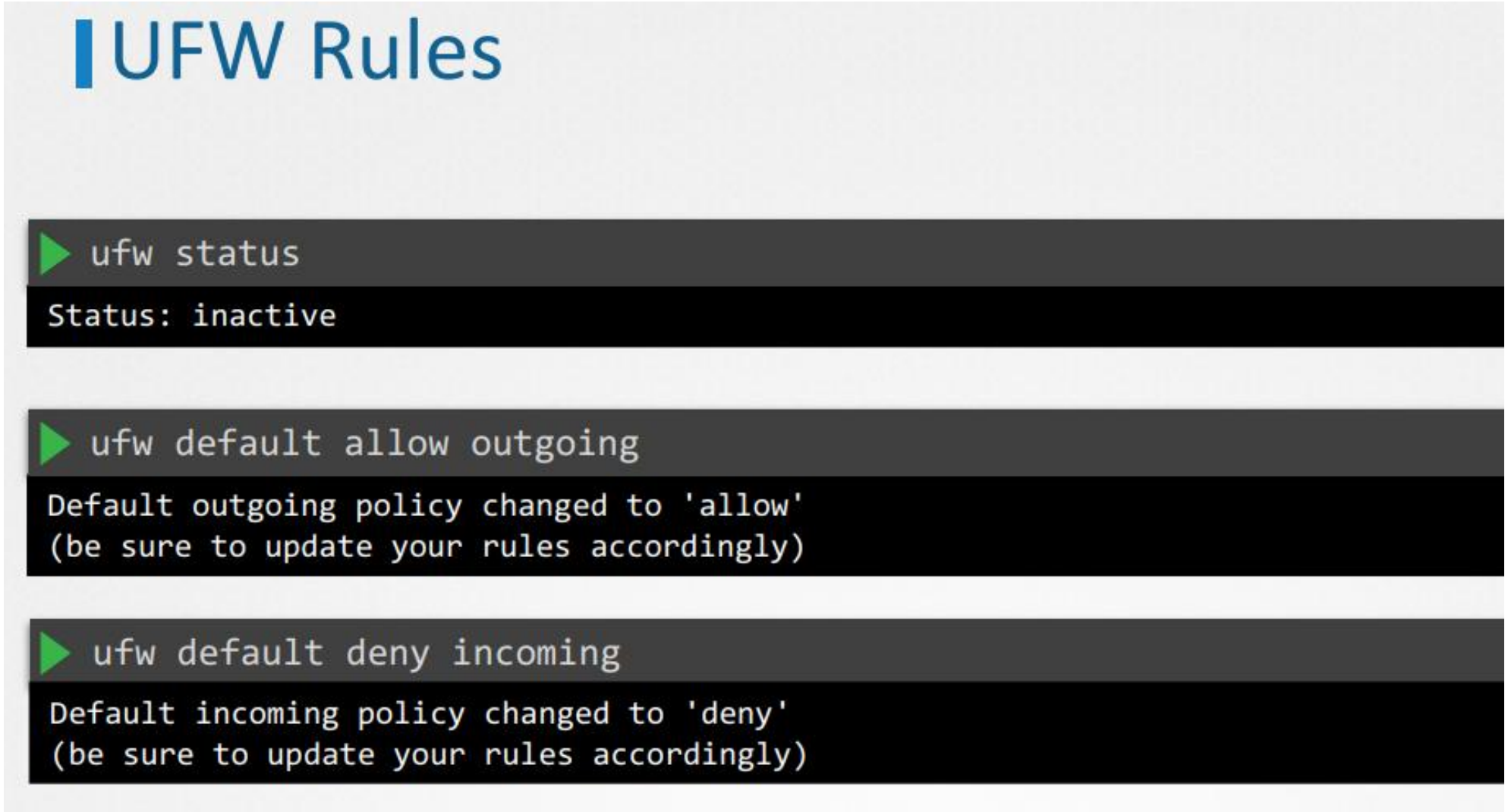
```
apt-get install ufw
```

```
systemctl enable ufw
```

```
Synchronizing state of ufw.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ufw
```

```
systemctl start ufw
```

UFW Firewall basics



UFW Firewall basics

UFW Rules

```
ufw allow from 172.16.238.5 to any port 22 proto tcp
Rules updated

ufw allow from 172.16.238.5 to any port 80 proto tcp
Rules updated

ufw allow from 172.16.100.0/28 to any port 80 proto tcp
Rules updated

ufw deny 8080
Rules updated
```

UFW Firewall basics

Enable UFW

```
ufw enable
```

```
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

```
ufw status
```

```
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       172.16.238.5
80/tcp                     ALLOW       172.16.238.5
80/tcp                     ALLOW       172.16.100.0/28
8080                       DENY        Anywhere
8080 (v6)                  DENY        Anywhere (v6)
```

UFW Firewall basics

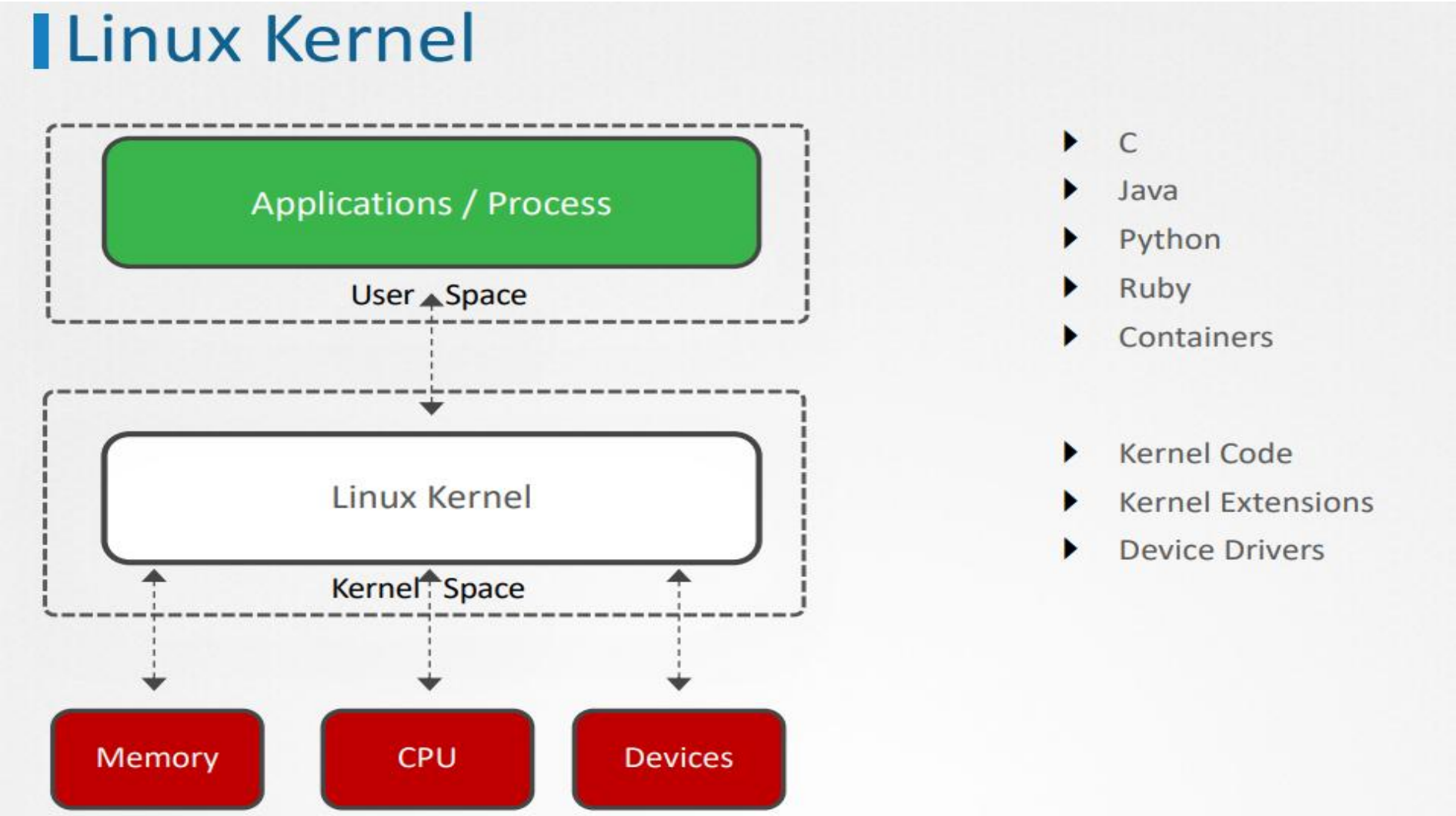
Delete Rules

```
ufw delete deny 8080
Rule deleted
Rule deleted (v6)
```

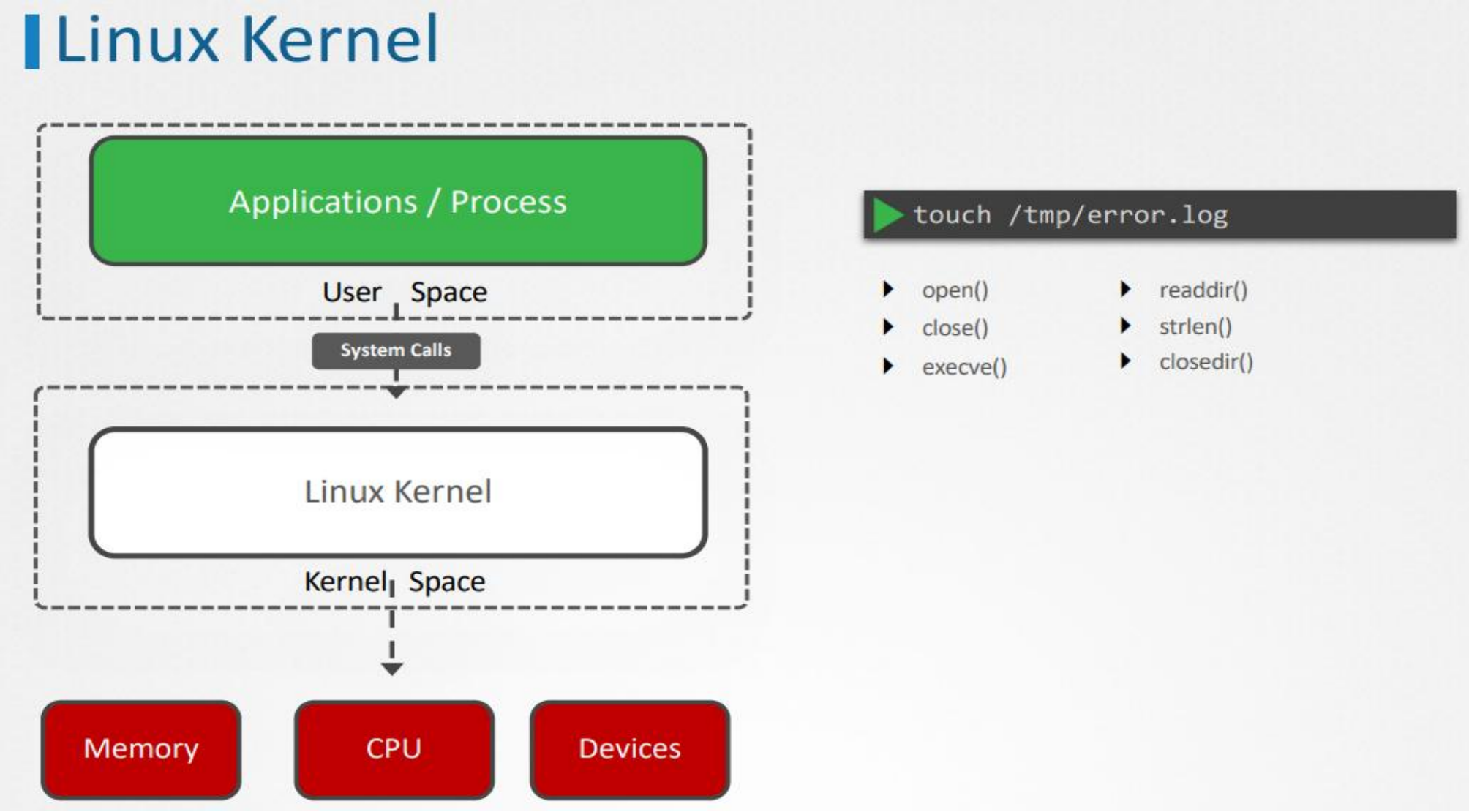
```
ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       172.16.238.5      --> 1
80/tcp                     ALLOW       172.16.238.5      --> 2
80/tcp                     ALLOW       172.16.100.0/28   --> 3
8080                       DENY        Anywhere          --> 4
8080 (v6)                  DENY        Anywhere (v6)     --> 5
```

```
ufw delete 5
Deleting:
 deny 8080
Proceed with operation (y|n)? y
Rule deleted (v6)
```

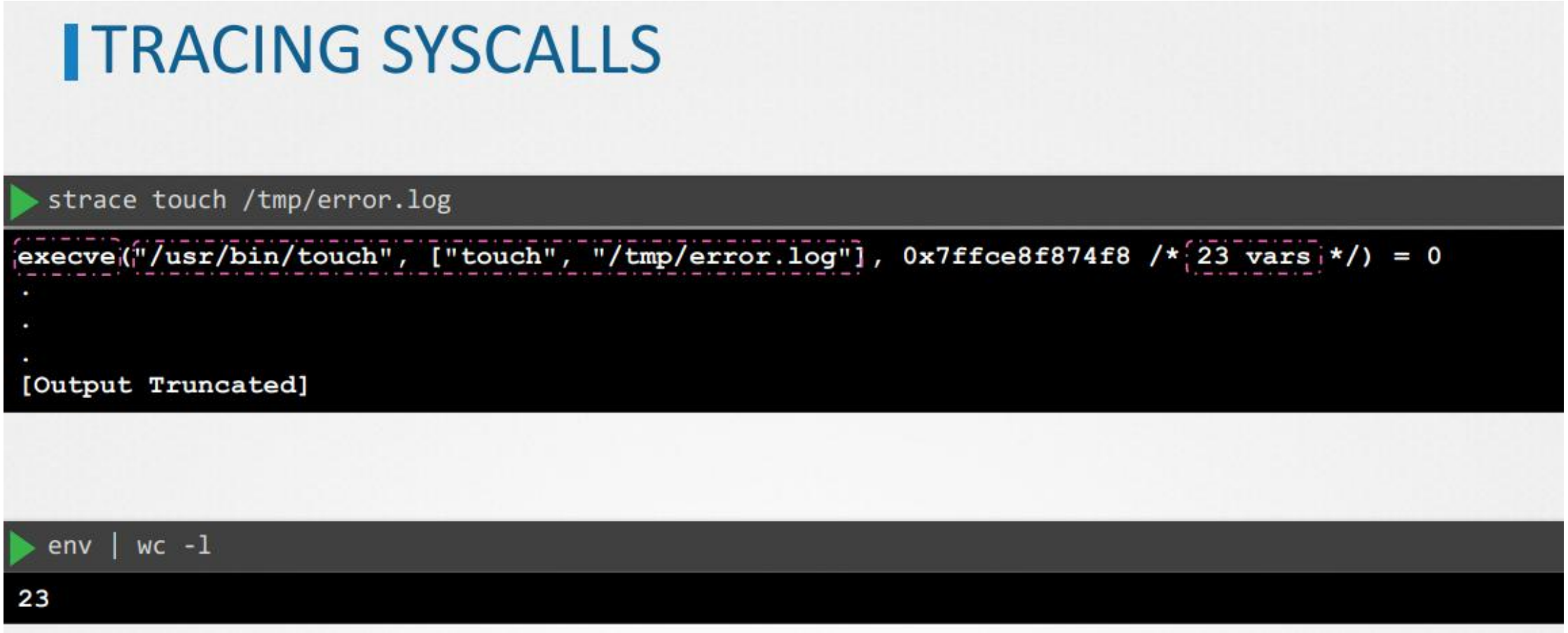
Linux syscalls



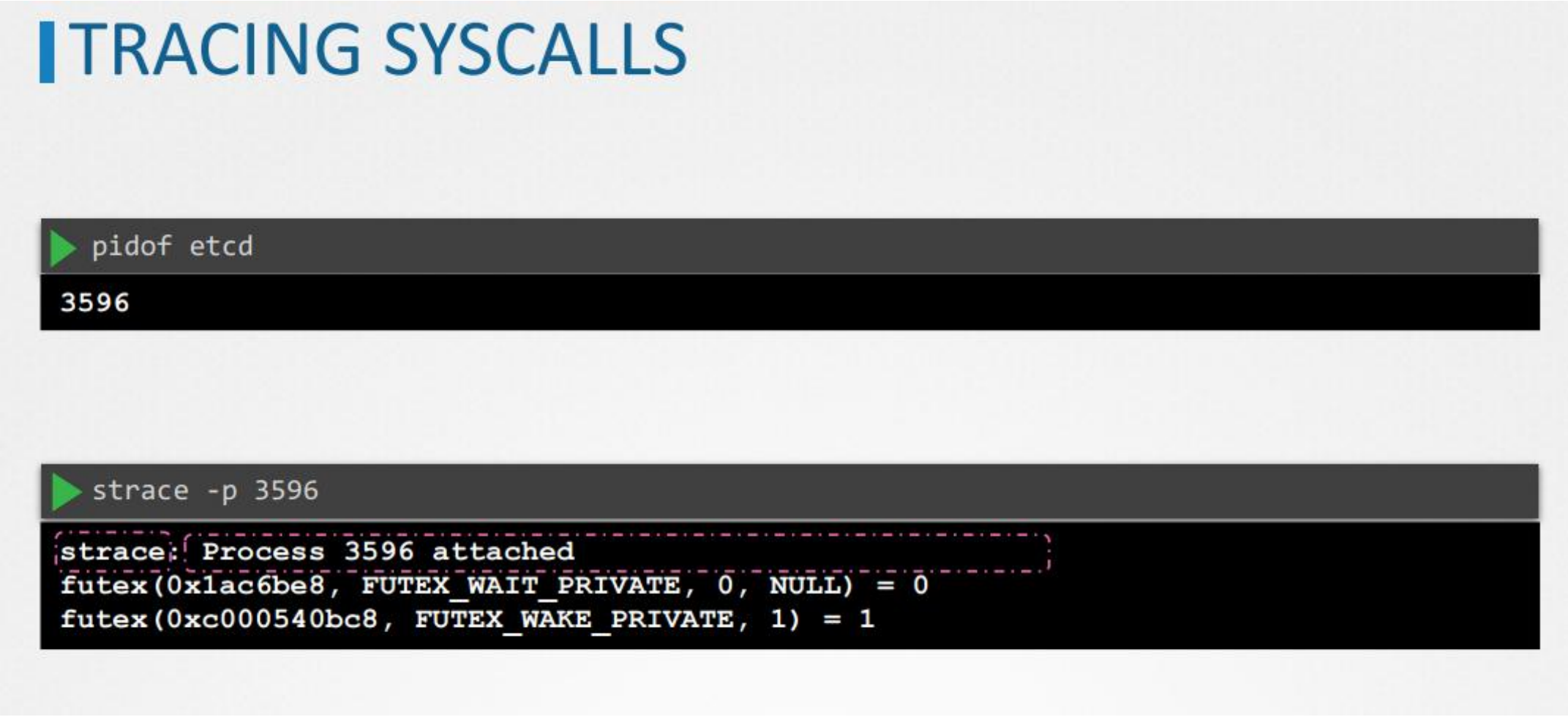
Linux syscalls



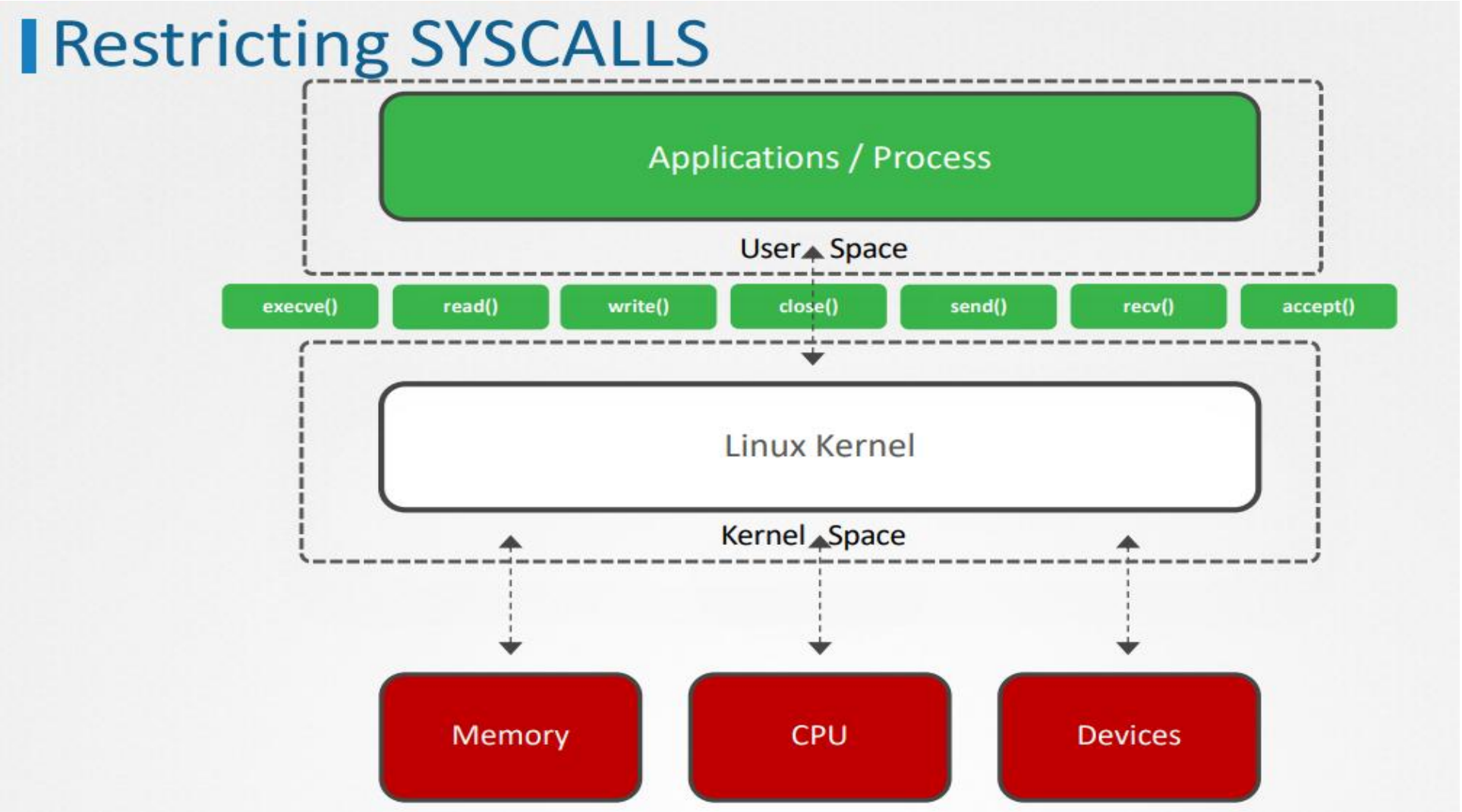
Linux syscalls



Linux syscalls



Restrict syscalls using seccomp



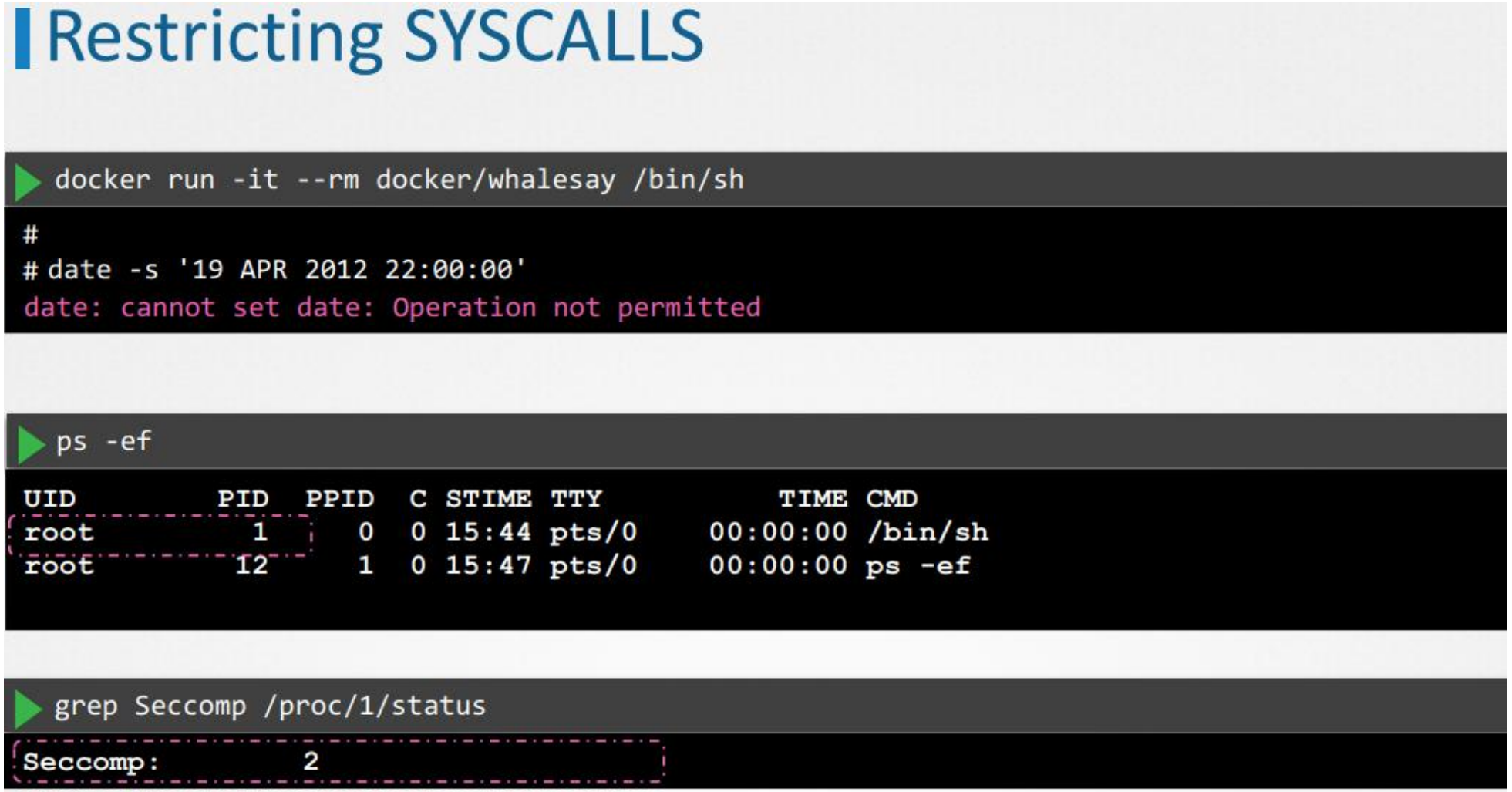
Restrict syscalls using seccomp

Restricting SYSCALLS

```
grep -i seccomp /boot/config-$(uname -r)
CONFIG_HAVE_ARCH_SECCOMP_FILTER=y
CONFIG_SECCOMP_FILTER=y
CONFIG_SECCOMP=y
```

```
docker run docker/whalesay cowsay  hello!
 _______
< hello! >
 -------
    \
     \
      \
                    ##         .
              ## ## ##        ==
           ## ## ## ##       ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \     \        __/
          \____\______/
```

Restrict syscalls using seccomp



Restrict syscalls using seccomp

Restricting SYSCALLS

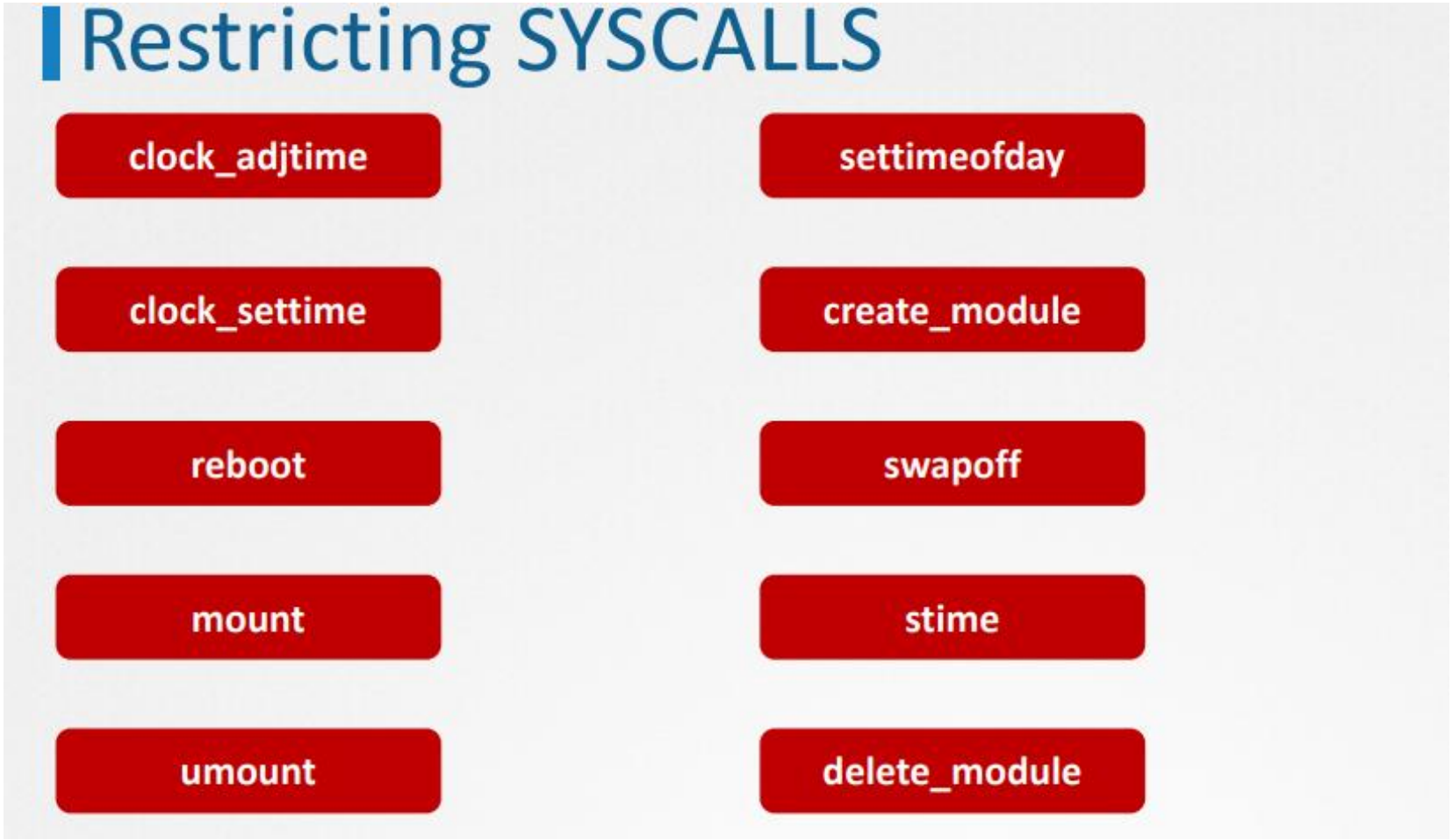
whitelist.json

```
{
    "defaultAction": "SCMP_ACT_ERRNO",
    "architectures": [
        "SCMP_ARCH_X86_64",
        "SCMP_ARCH_X86",
        "SCMP_ARCH_X32"
    ],
    "syscalls": [
        {
            "names": [
                "<syscall-1>",
                "<syscall-2>",
                "<syscall-3>"
            ],
            "action": "SCMP_ACT_ALLOW"
        }
    ]
}
```

blacklist.json

```
{
    "defaultAction": "SCMP_ACT_ALLOW",
    "architectures": [
        "SCMP_ARCH_X86_64",
        "SCMP_ARCH_X86",
        "SCMP_ARCH_X32"
    ],
    "syscalls": [
        {
            "names": [
                "<syscall-1>",
                "<syscall-2>",
                "<syscall-3>"
            ],
            "action": "SCMP_ACT_ERRNO"
        }
    ]
}
```

Restrict syscalls using seccomp



Restrict syscalls using seccomp

Restricting SYSCALLS

custom.json

```
{
    "defaultAction": "SCMP_ACT_ERRNO",
    "architectures": [
        "SCMP_ARCH_X86_64",
        "SCMP_ARCH_X86",
        "SCMP_ARCH_X32"
    ],
    "syscalls": [
        {
            "names": [
                "arch_prctl",
                "brk",
                "capget",
                "capset",

                "close",
                "execve",

                .
                .
                "clone"
            ],
            "action": "SCMP_ACT_ALLOW"
        }
```

```
docker run -it --rm --security-opt seccomp=/root/custom.json \
docker/whalesay /bin/sh
/ #
/ # mkdir test
mkdir: can't create directory 'test': Operation not permitted
```

Implement seccomp in kubernetes

Seccomp in Kubernetes

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name:  test-audit
spec:
  securityContext:
    seccompProfile:
      type: Localhost
      localhostProfile: <path to the custom JSON file>
  containers:
  - command: ["bash", "-c", "echo 'I just made some syscalls' && sleep 100"]
    image:  ubuntu
    name: ubuntu
    securityContext:
      allowPrivilegeEscalation: false
```

/var/lib/kubelet/seccomp

Implement seccomp in kubernetes

Seccomp in Kubernetes

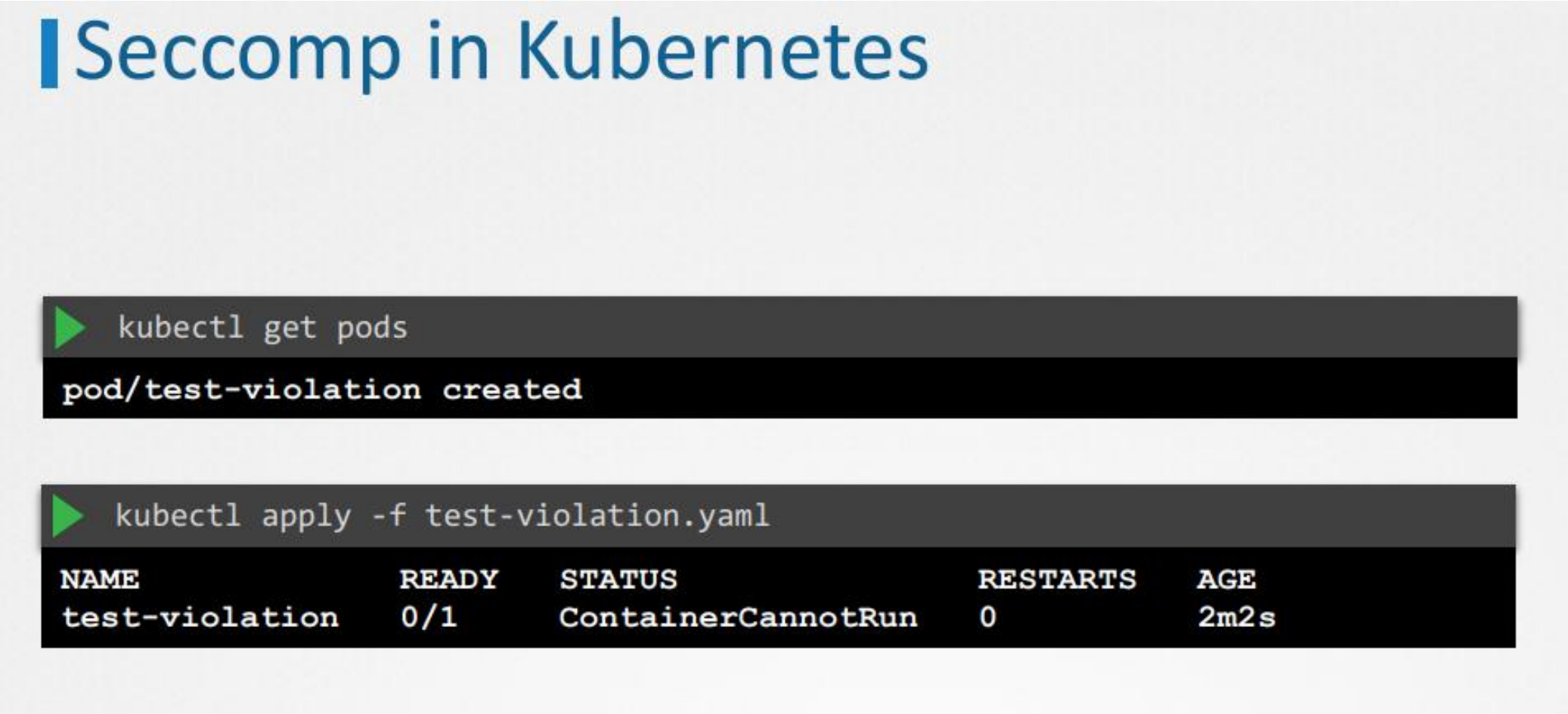
/var/lib/kubelet/seccomp/profiles /violation.json

```
{
    "defaultAction": "SCMP_ACT_ERRNO"
}
```

test-violation.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: test-violation
spec:
  securityContext:
    seccompProfile:
      type: Localhost
      localhostProfile: profiles/violation.json
  containers:
  - command: ["bash", "-c", "echo 'I just made some syscalls' && sleep 100"]
    image: ubuntu
    name: ubuntu
  restartPolicy: Never
```

Implement seccomp in kubernetes



AppArmor

```
systemctl status apparmor
```

```
● apparmor.service - AppArmor initialization
   Loaded: loaded (/lib/systemd/system/apparmor.service; enabled; vendor preset:
enabled)
   Active: active (exited) since Mon 2021-03-22 03:12:41 UTC; 2min 29s ago
     Docs: man:apparmor(7)
           http://wiki.apparmor.net/
 Main PID: 313 (code=exited, status=0/SUCCESS)
    Tasks: 0 (limit: 4678)
   CGroup: /system.slice/apparmor.service
```

AppArmor

```
cat /sys/module/apparmor/parameters/enabled
Y
```

```
cat /sys/kernel/security/apparmor/profiles
docker-default (enforce)
/usr/sbin/tcpdump (enforce)
/usr/sbin/ntpd (enforce)
/usr/lib/snapd/snap-confine (enforce)
/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
(enforce)
/usr/lib/connman/scripts/dhclient-script (enforce)
/usr/lib/NetworkManager/nm-dhcp-helper (enforce)
/usr/lib/NetworkManager/nm-dhcp-client.action (enforce)
/sbin/dhclient (enforce)
```

AppArmor

apparmor-deny-write

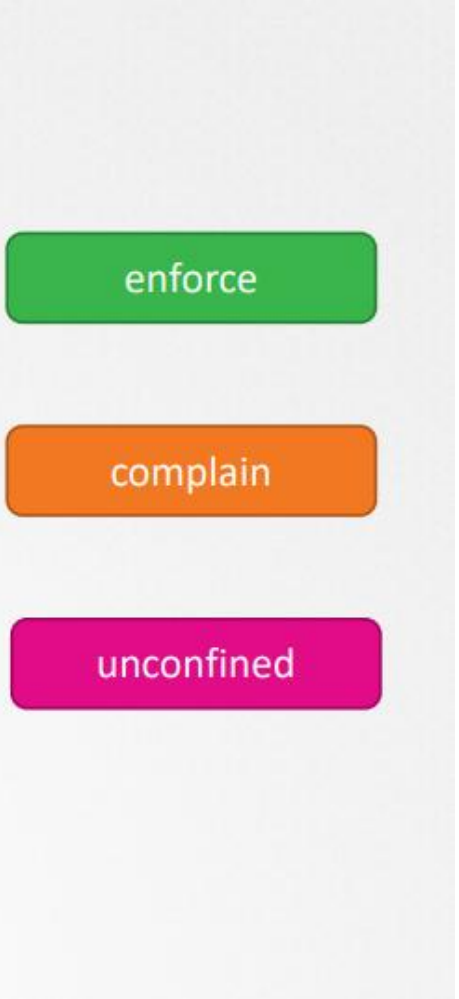
```
profile apparmor-deny-write flags=(attach_disconnected) {
  file,
  # Deny all file writes.
  deny /** w,
}
```

apparmor-deny-proc-write

```
profile apparmor-deny-proc-write flags=(attach_disconnected) {
  file,
  # Deny all file writes to /proc.
  deny /proc/* w,
}
```

AppArmor

```
aa-status
apparmor module is loaded.
12 profiles are loaded.
12 profiles are in enforce mode.
   /sbin/dhclient
   /usr/bin/man
   /usr/lib/NetworkManager/nm-dhcp-client.action
   /usr/lib/NetworkManager/nm-dhcp-helper
.
.
/usr/sbin/tcpdump
   docker-default
   man_filter
   man_groff
0 profiles are in complain mode.
11 processes have profiles defined.
11 processes are in enforce mode.
   /sbin/dhclient (621)
   docker-default (3970)
   docker-default (4025)
docker-default (9853)
   docker-default (9964)
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```



Security Context

Security Context

Security Context

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-pod
spec:
  securityContext:
    runAsUser: 1000

  containers:
      - name: ubuntu
        image: ubuntu
        command: ["sleep", "3600"]
```

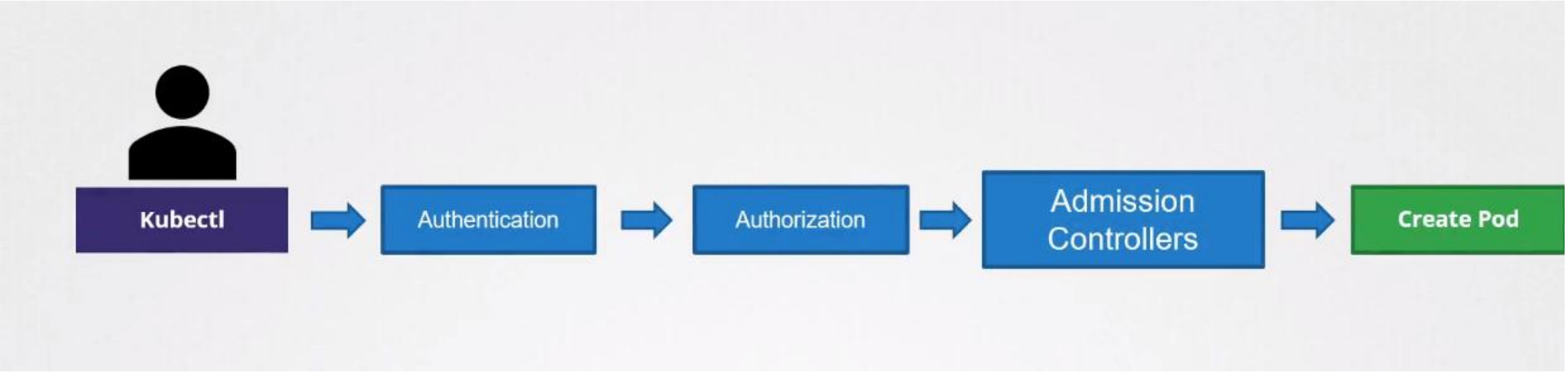
Security Context

Security Context

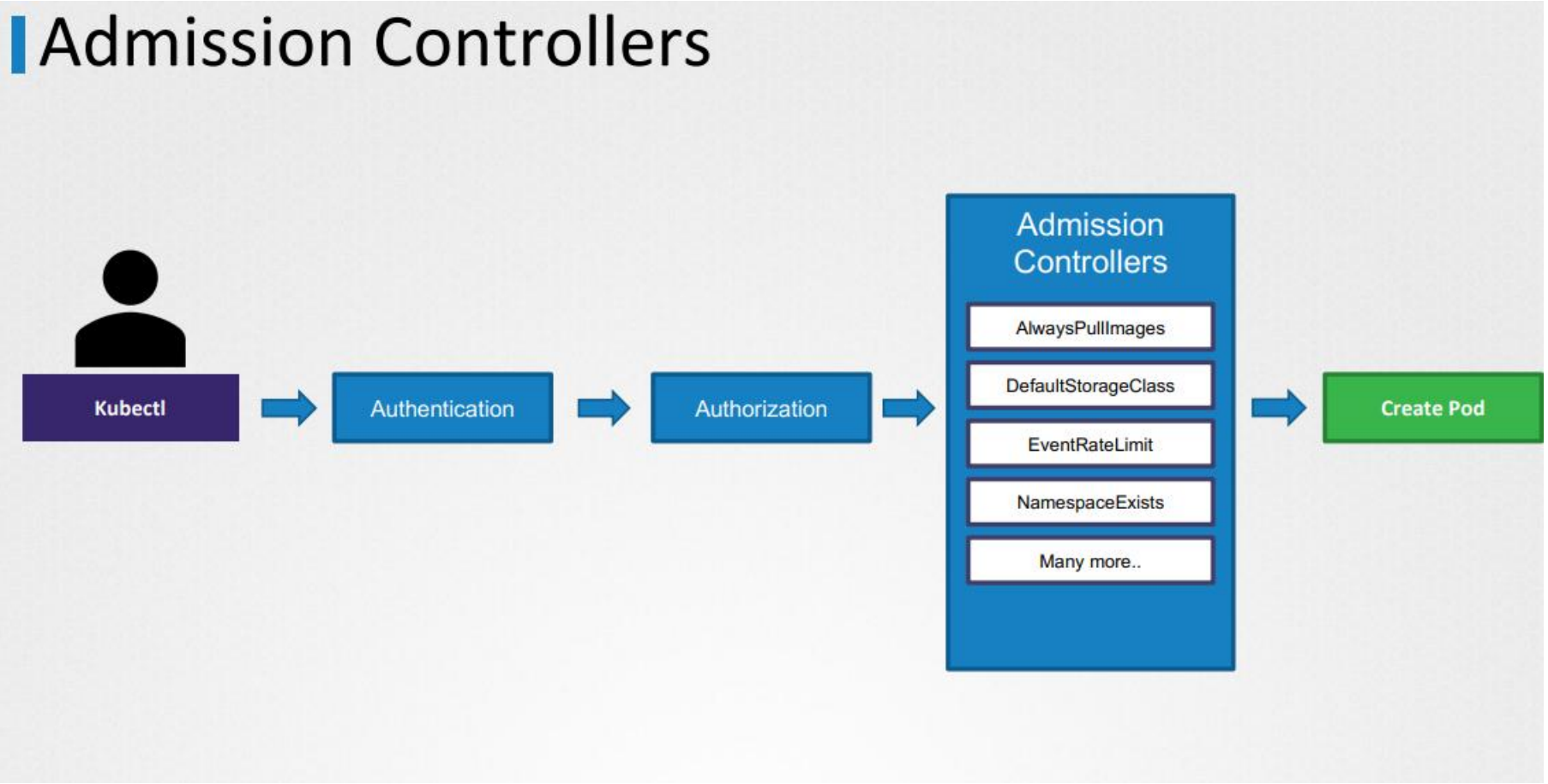
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-pod
spec:
  containers:
      - name: ubuntu
        image: ubuntu
        command: ["sleep", "3600"]
        securityContext:
          runAsUser: 1000
          capabilities:
              add: ["MAC_ADMIN"]
```

Note: Capabilities are only supported at the container level and not at the POD level

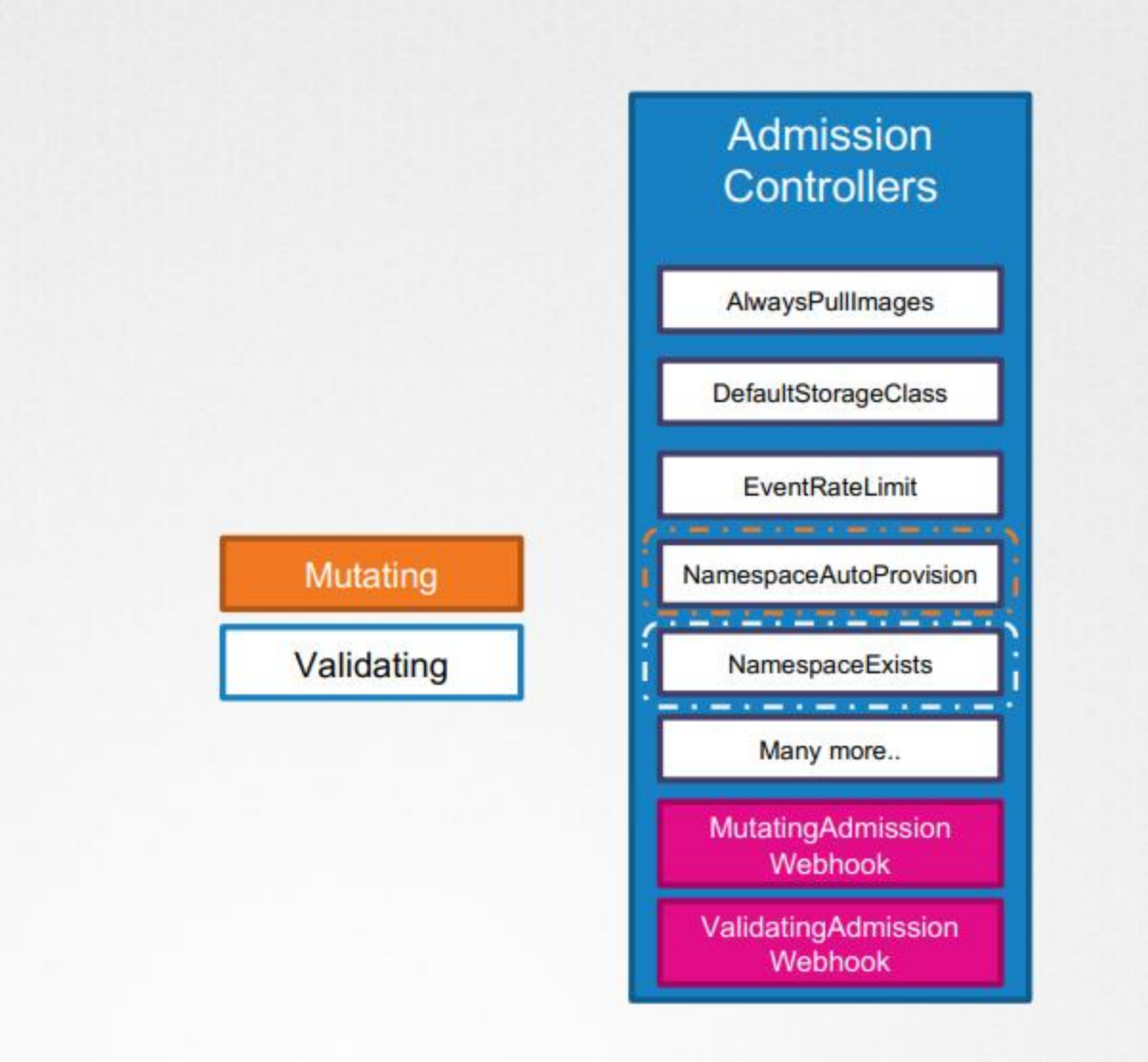
Admission Controller



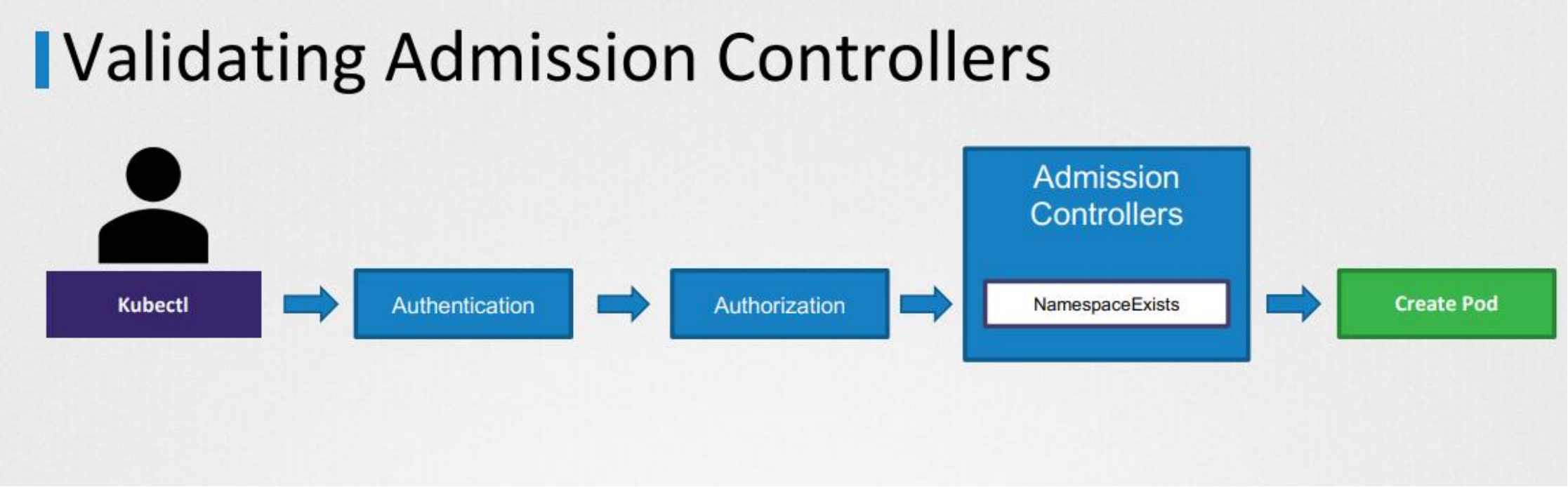
Admission Controller



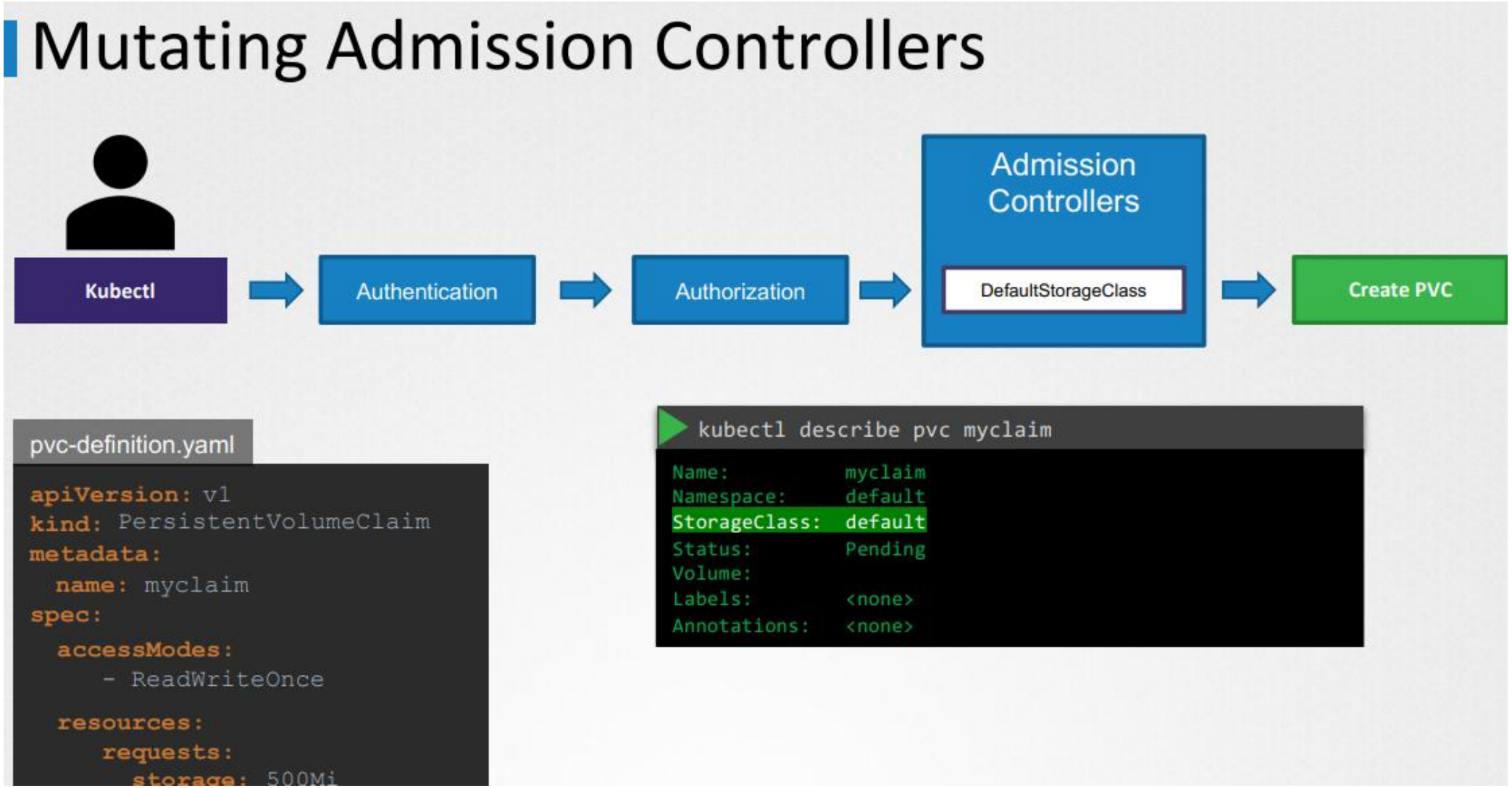
Mutating and Validating Admission Controller



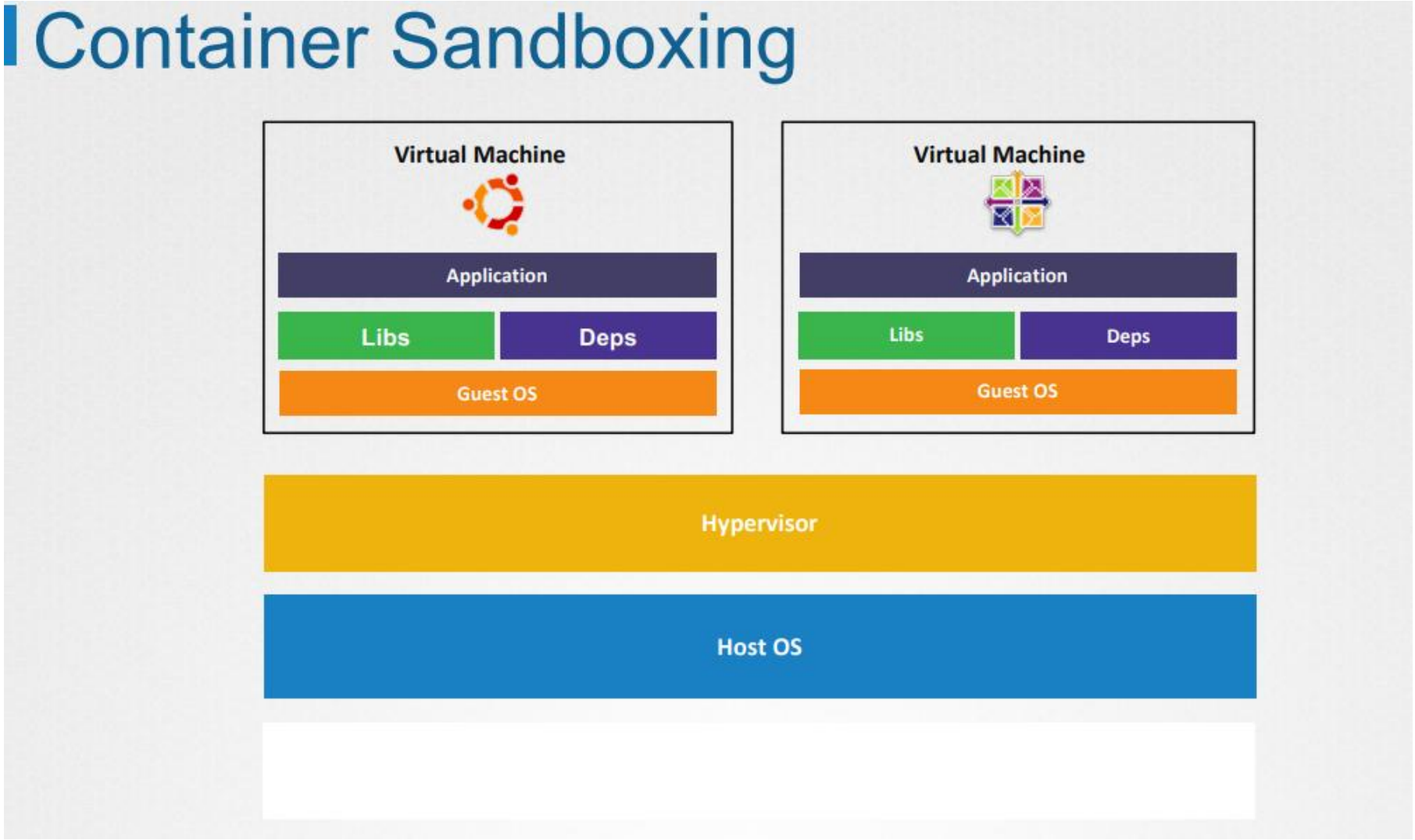
Mutating and Validating Admission Controller



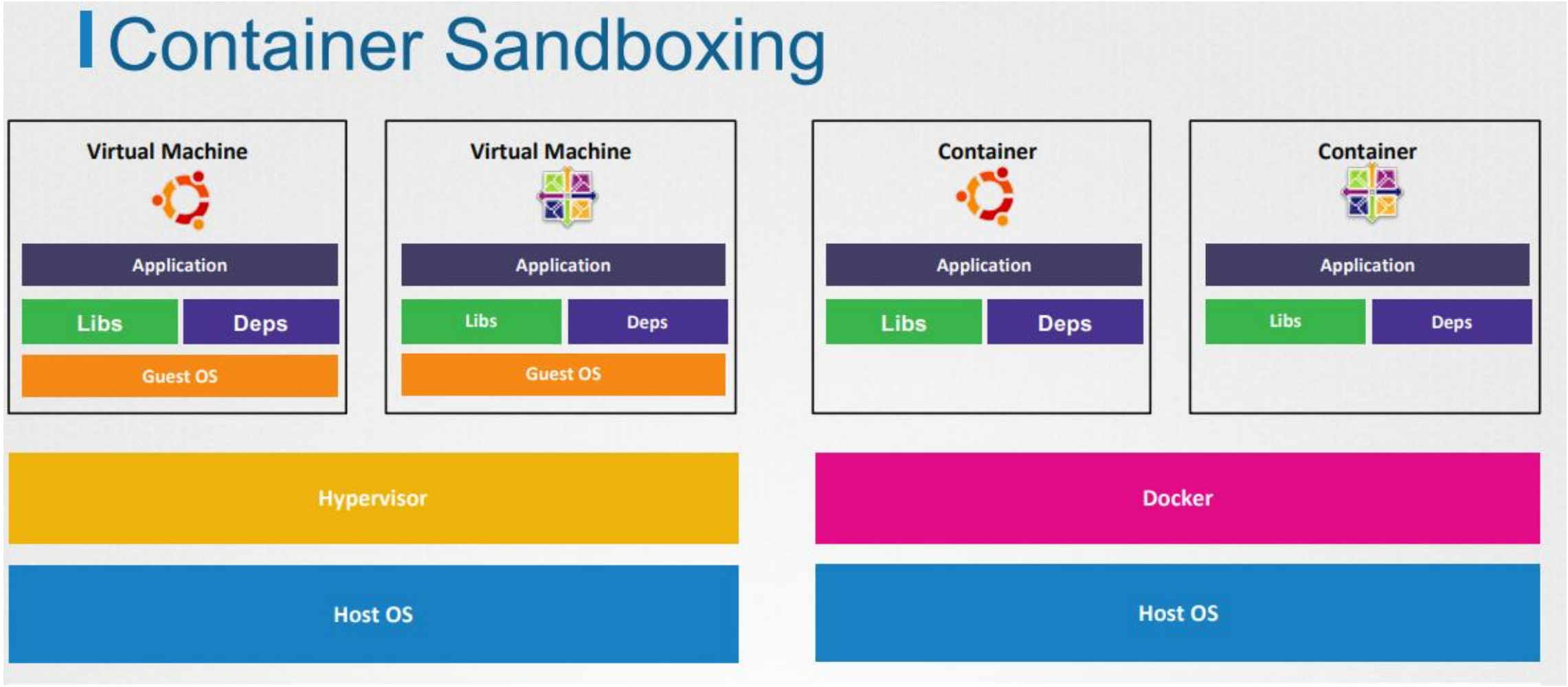
Mutating and Validating Admission Controller



Container Sandboxing



Container Sandboxing



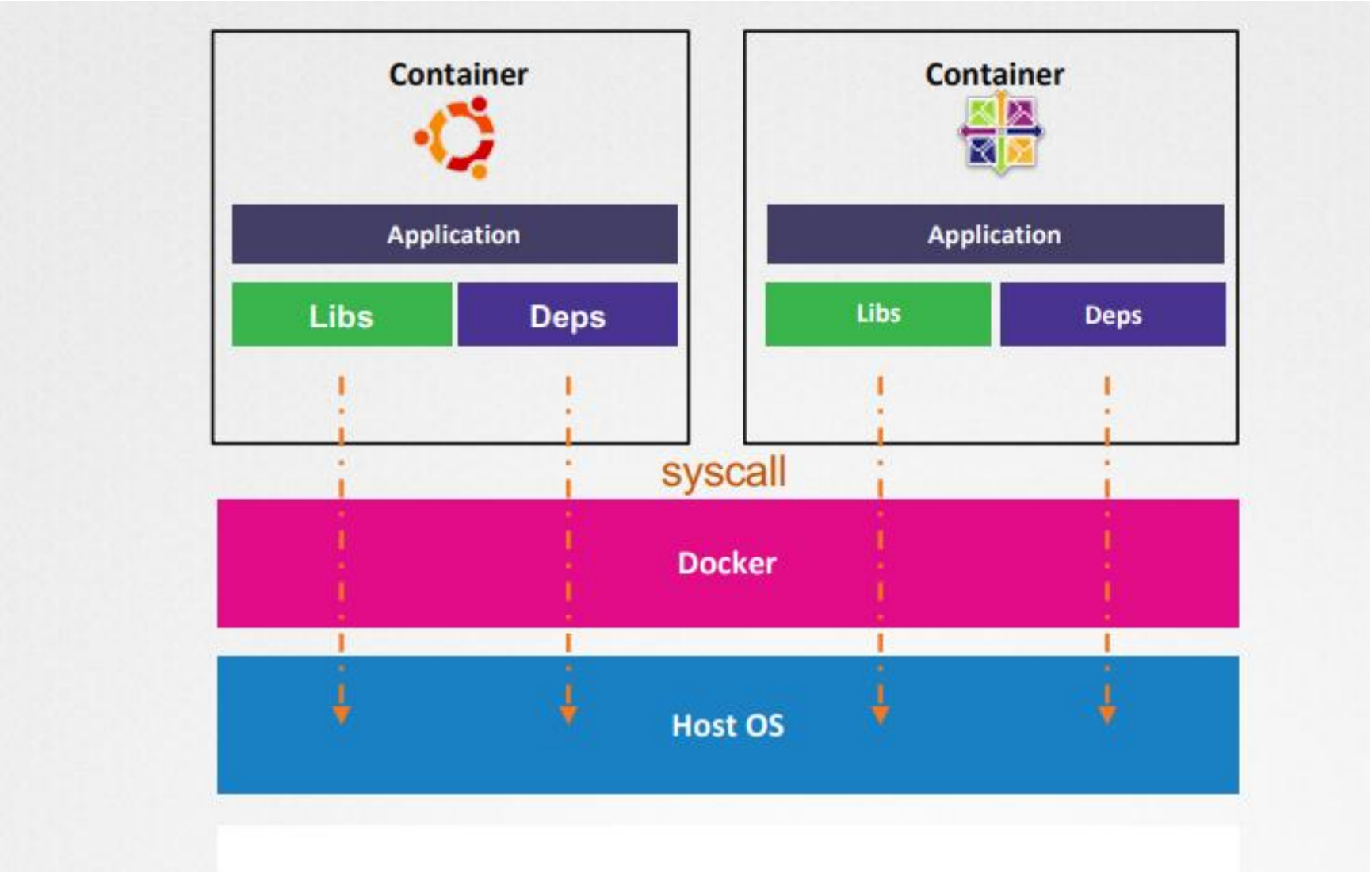
Container Sandboxing

```
root@ubuntu-server:~# docker run -d --name sleeping-container  busybox sleep 1000
e2fd5090c9a51eb7cc91a466cf2e18c5468871f84adbb55c2e6c1cf4ea0028a8
```

```
root@ubuntu-server:~# docker exec -ti sleeping-container  ps -ef
PID   USER     TIME  COMMAND
    1 root      0:00 sleep 1000
   11 root      0:00 ps -ef
```

```
root@ubuntu-server:~#  ps -ef | grep sleep | grep -vi grep
root      7902  7871  0 21:39 ?        00:00:00 sleep 1000
```

Container Sandboxing



Container Sandboxing

Seccomp

custom.json

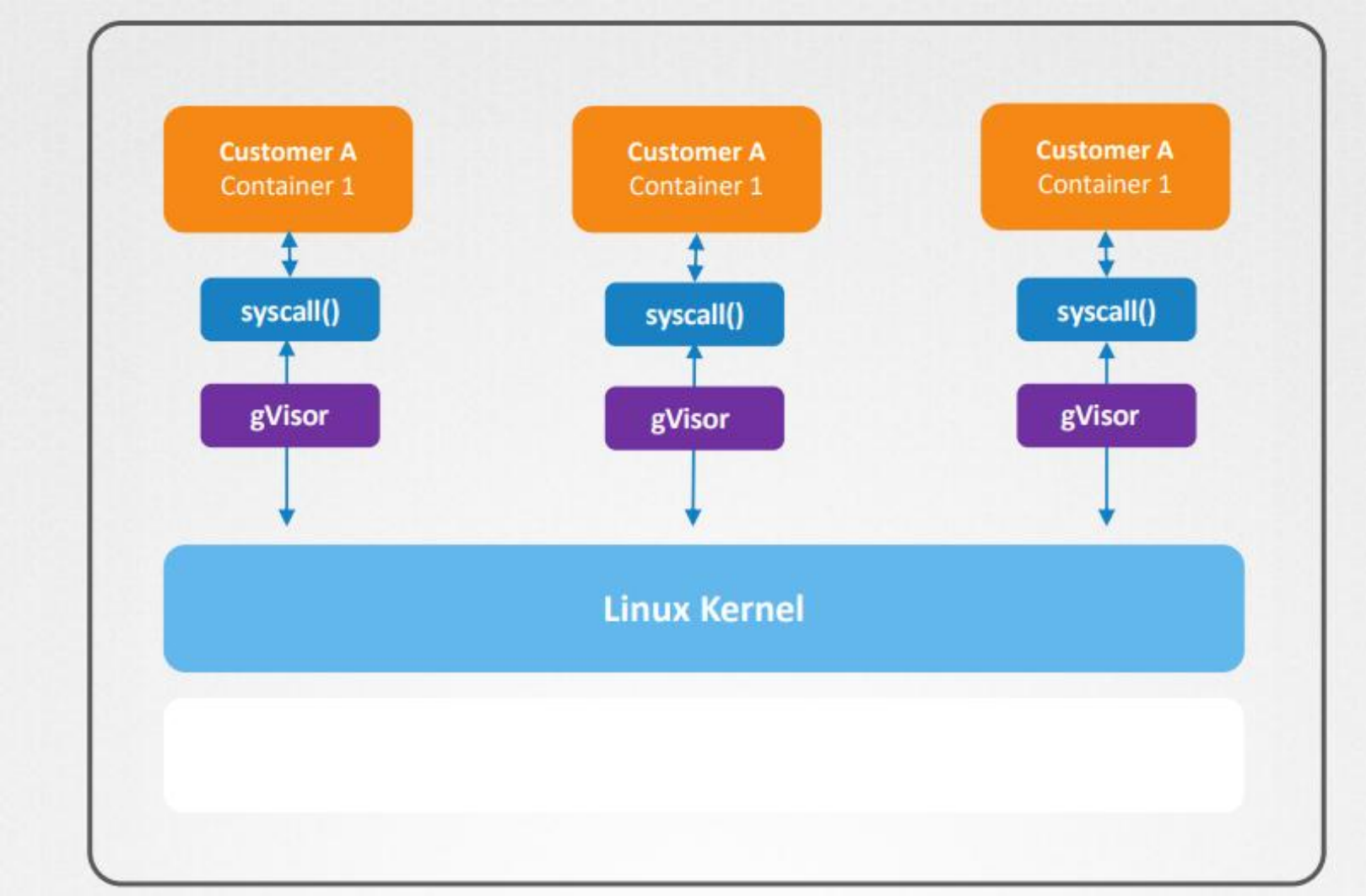
```
{
    "defaultAction": "SCMP_ACT_ERRNO",
    "architectures": [
        "SCMP_ARCH_X86_64",
        "SCMP_ARCH_X86",
        "SCMP_ARCH_X32"
    ],
    "syscalls": [
        {
            "names": [
                "execve",
                "brk",
                "access",
                "capset",
                "clone"
            ],
            "action": "SCMP_ACT_ALLOW"
        }
    ]
}
```

AppArmor

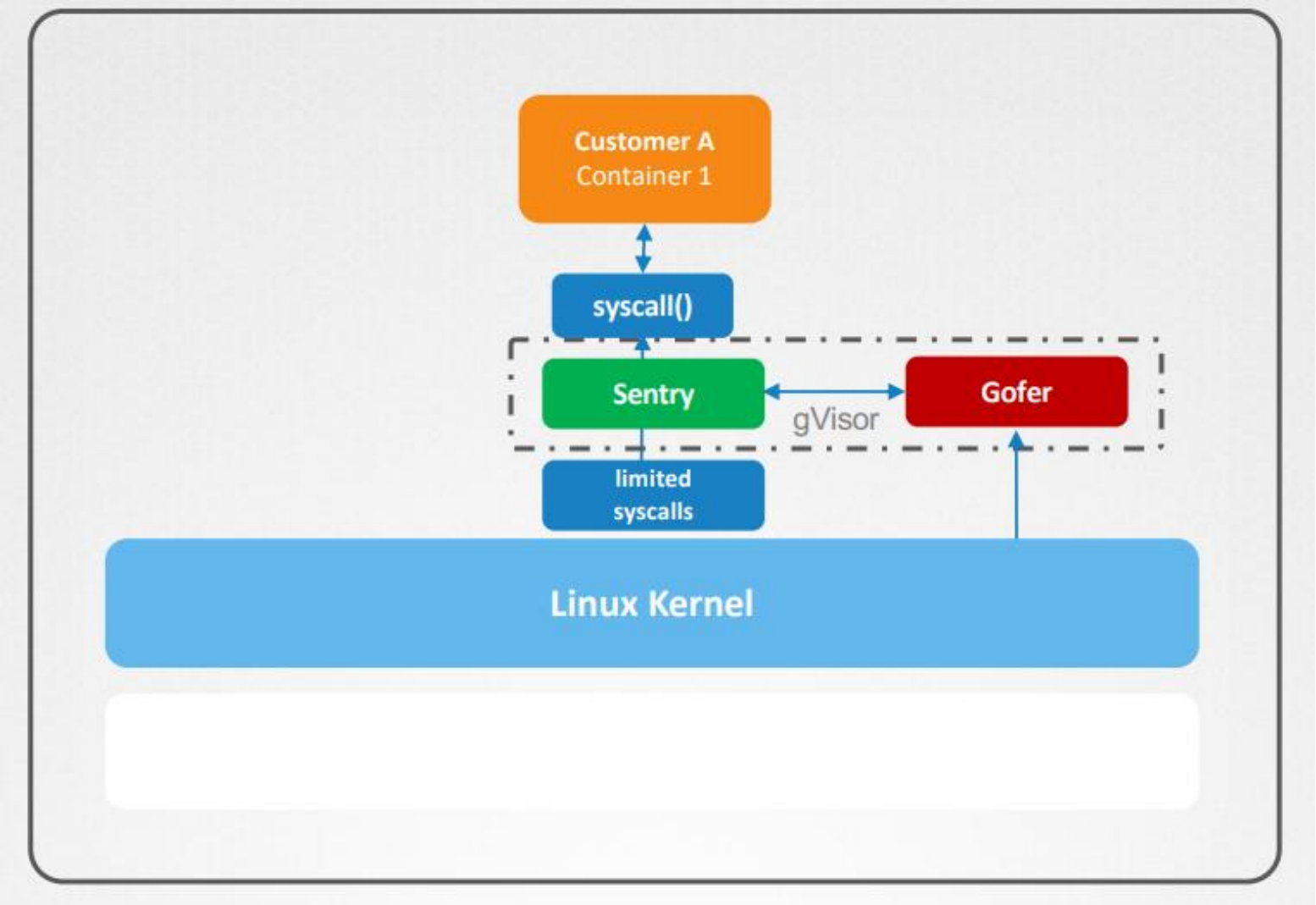
k8s-apparmor-example-deny-proc-write

```
profile apparmor-deny-write flags=(attach_disconnected) {
  file,
  # Deny all file writes to /proc.
  deny /proc/* w,
}
```

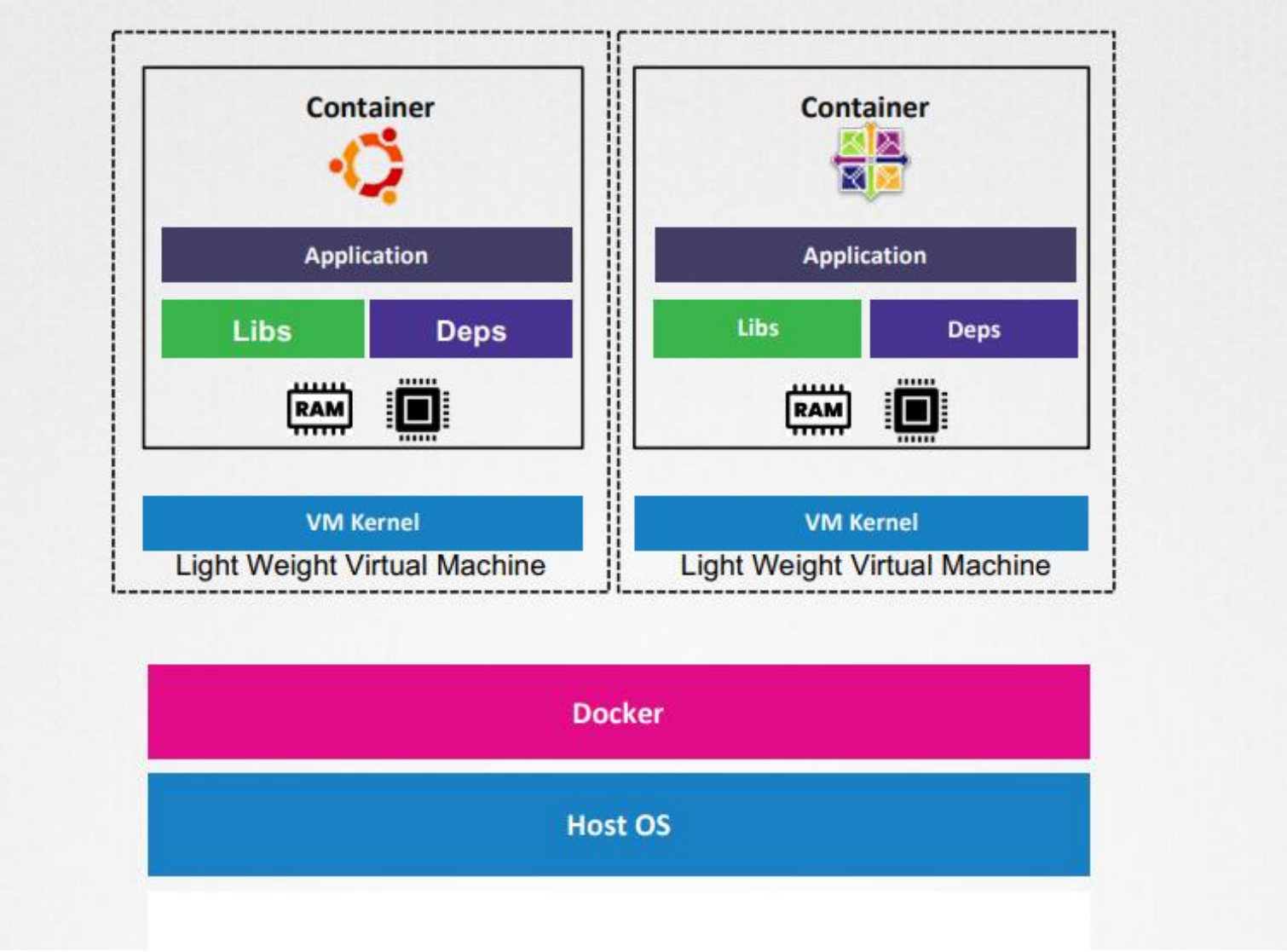
Container Sandboxing - gVisor



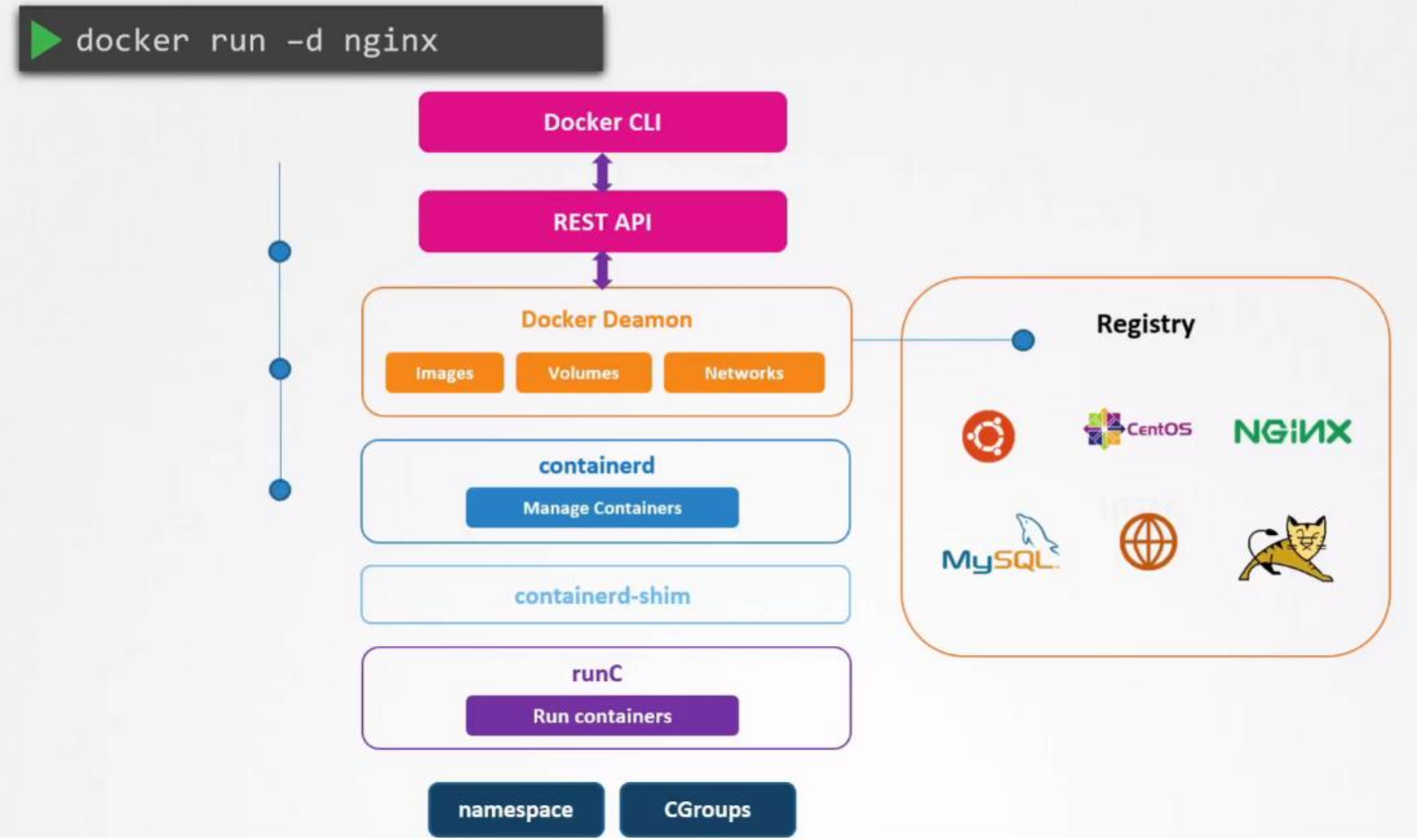
Container Sandboxing - gVisor



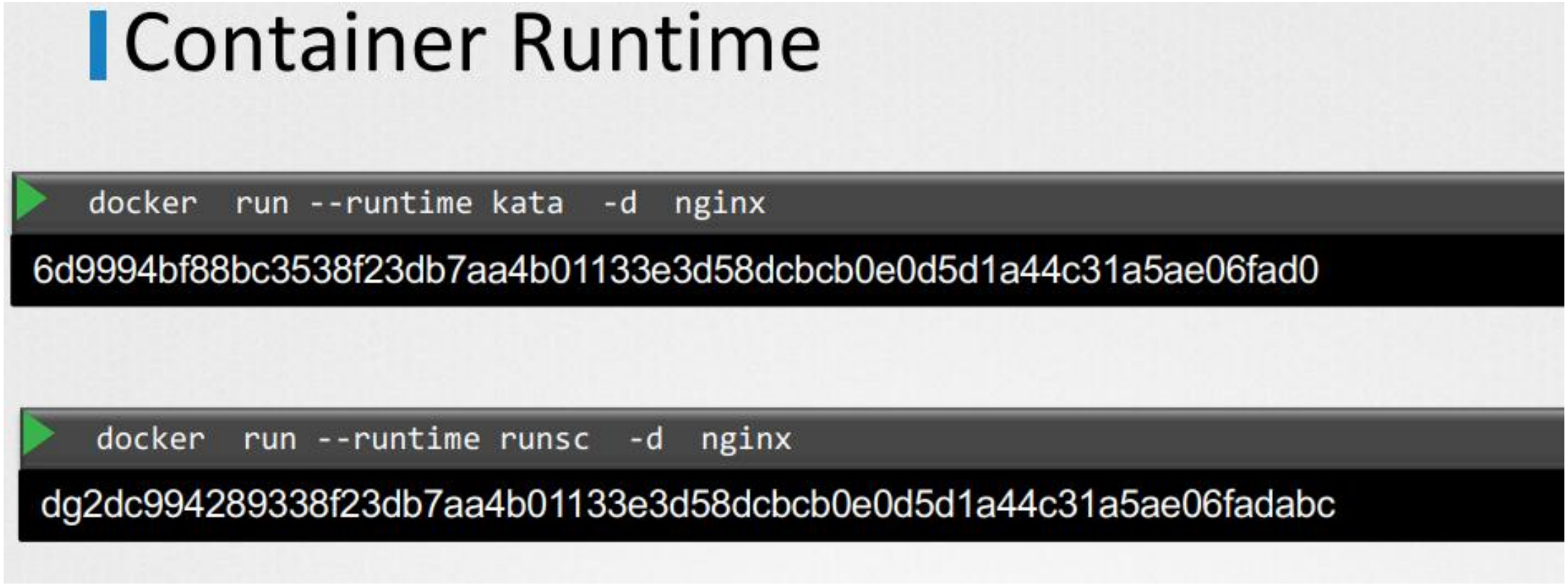
Container Sandboxing – Kata Containers



Runtime Classes



Runtime Classes



Runtime Classes in Kubernetes

gvisor.yaml

```
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:
  name: gvisor
handler: runsc
```

```
kubectl create -f gvisor.yaml
runtimeclass.node.k8s.io/gvisor created
```

Runtime	Handlers
gVisor	runsc
Kata	kata

Runtime Classes in Kubernetes

gvisor.yaml

```
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:
  name: gvisor
handler: runsc
```

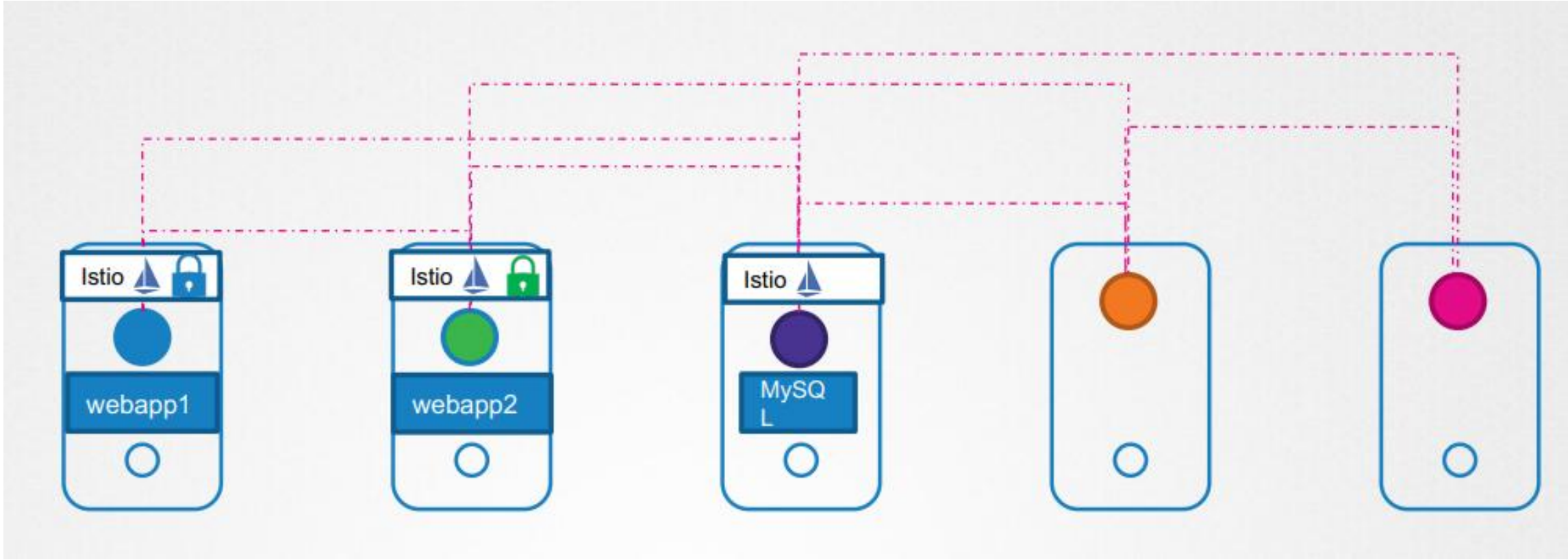
```
kubectl create -f gvisor.yaml
runtimeclass.node.k8s.io/gvisor created
```

gvisor-nginx.yaml

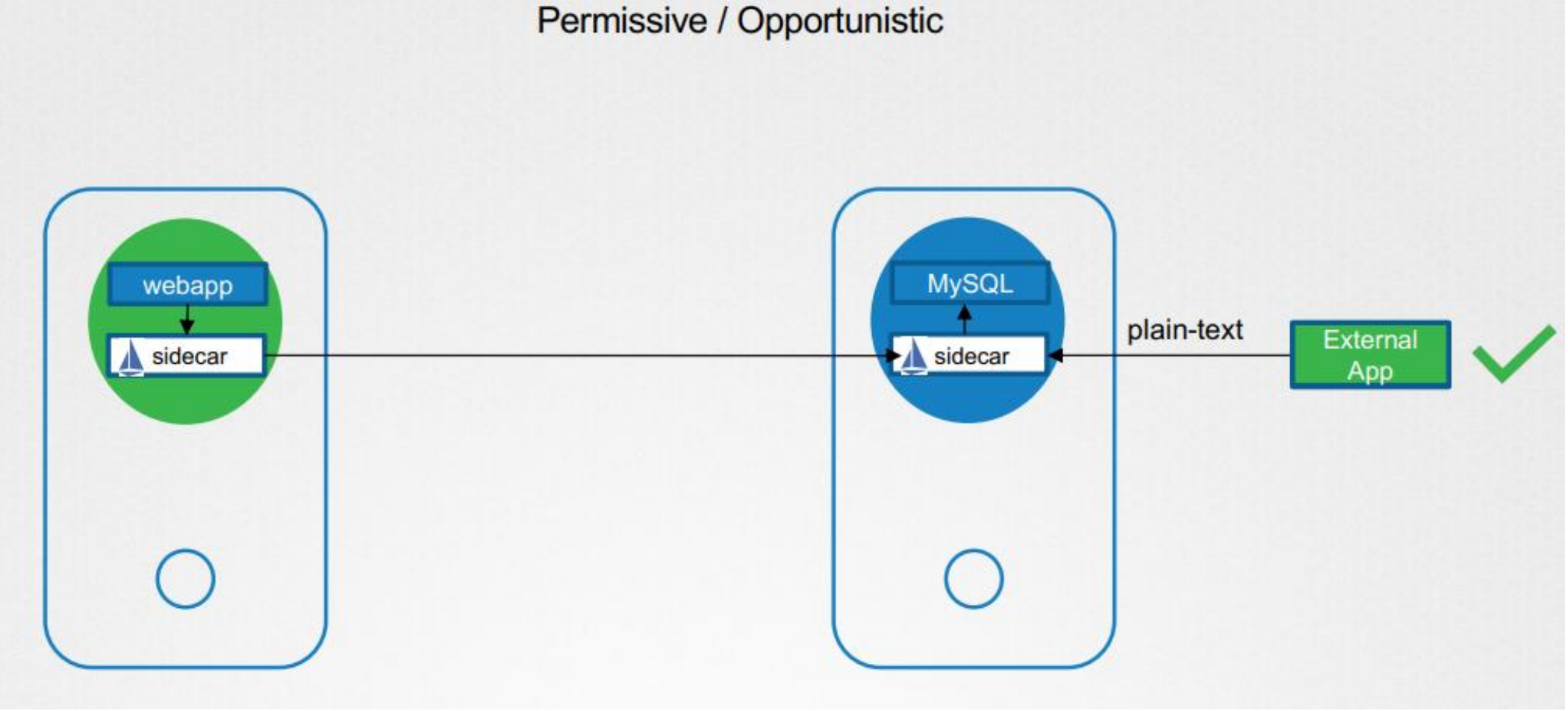
```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  runtimeClassName: gvisor
  containers:
  - image: nginx
    name: nginx
```

```
kubectl create -f gvisor-nginx.yaml
pod/nginx created
```

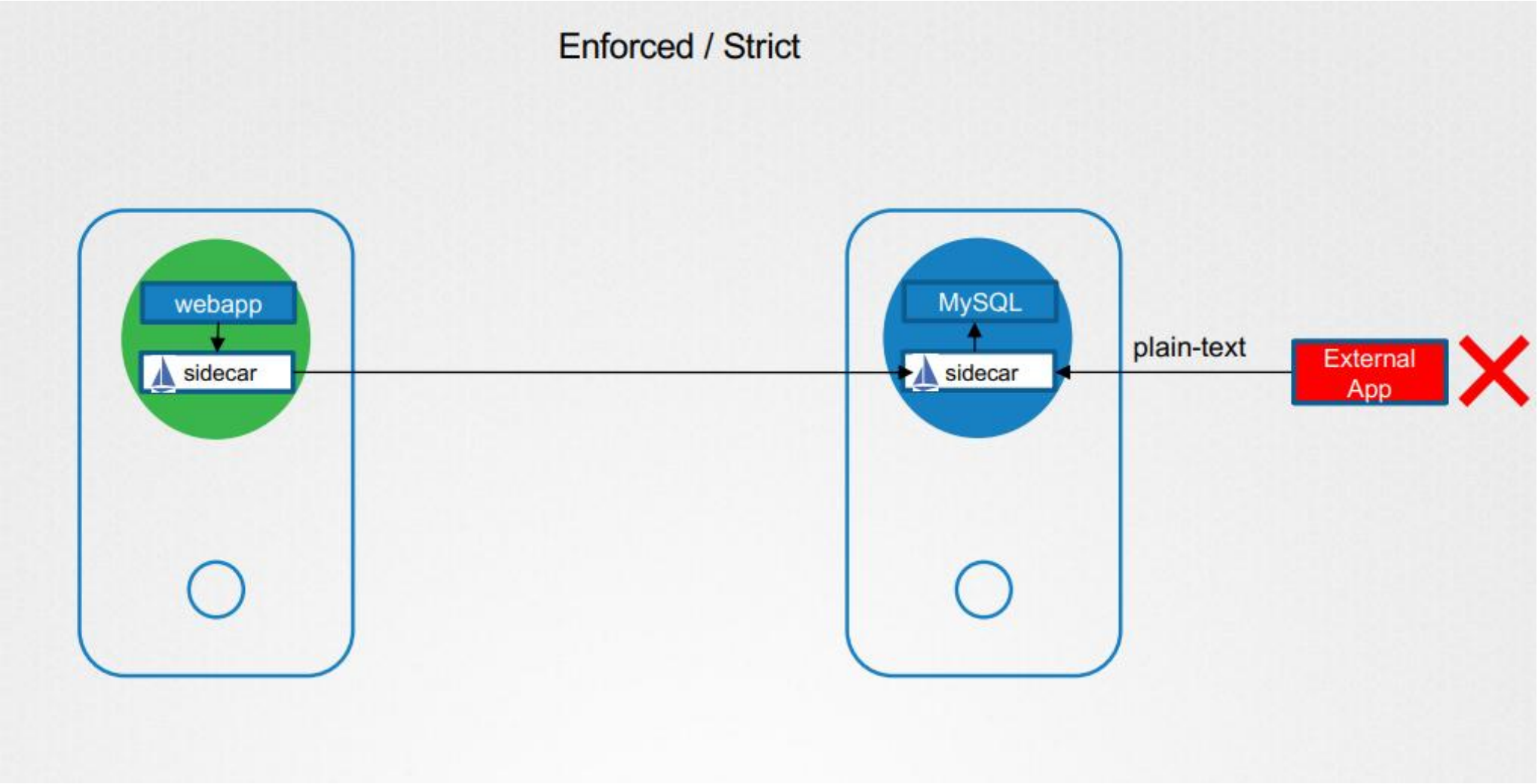
mTLS to secure POD-POD Communication



mTLS to secure POD-POD Communication



mTLS to secure POD-POD Communication



Minimize Base Image Footprint

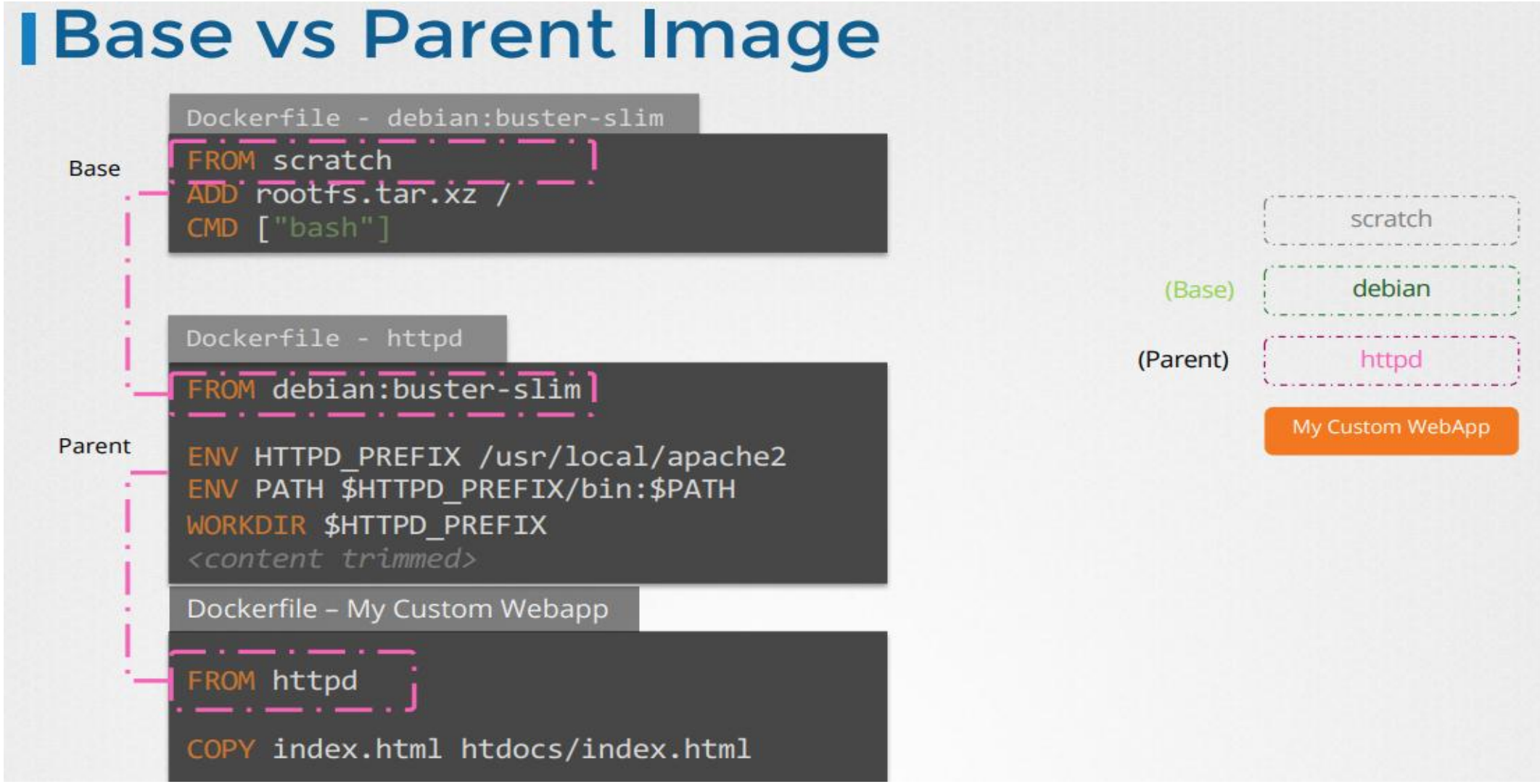


Image Security

nginx-pod.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
```

Image Security

```
image: docker.io/nginx/nginx
```

Registry

User/
Account

Image/
Repository

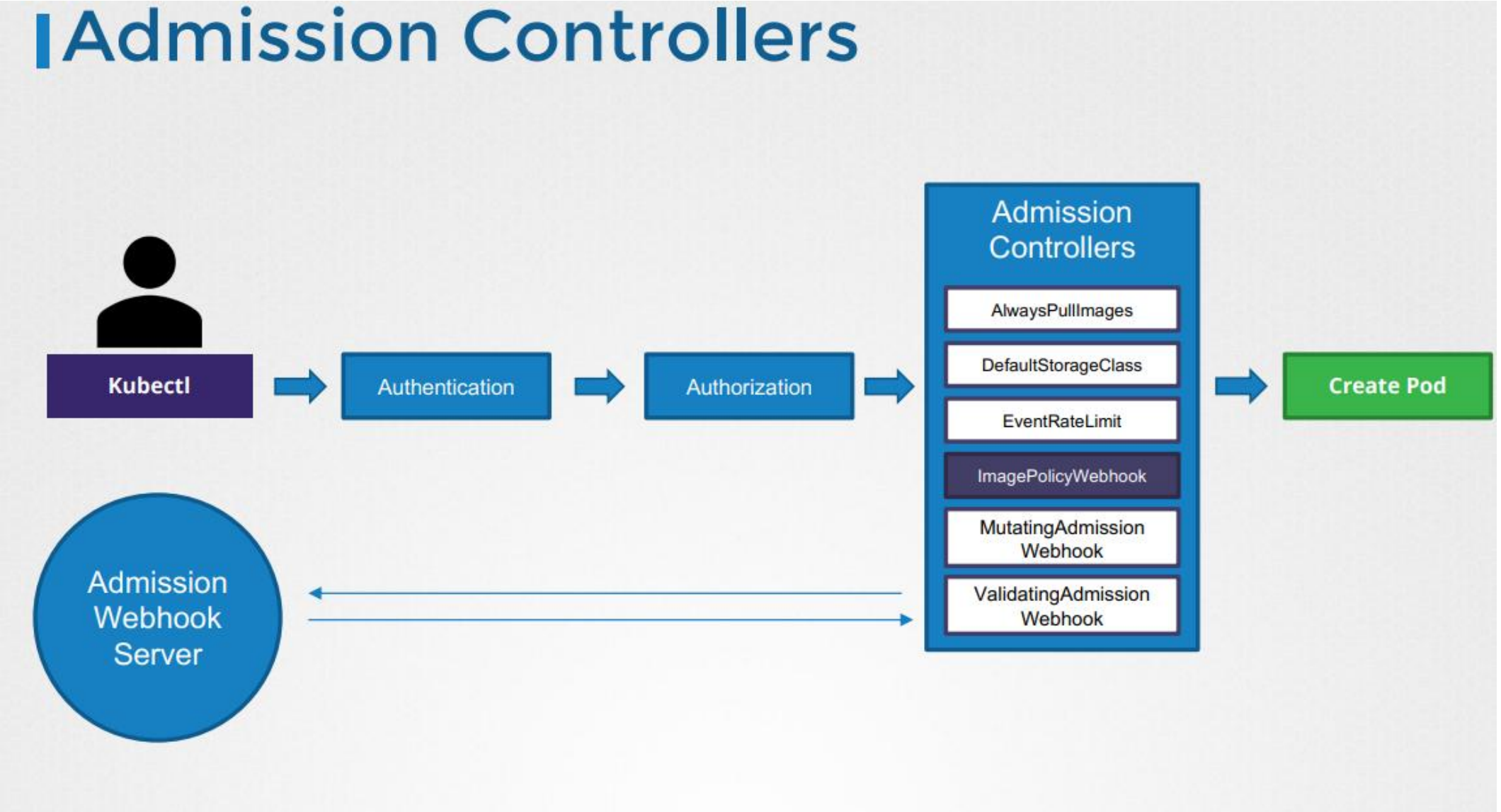
```
gcr.io/ kubernetes-e2e-test-images/dnsutils
```

Image Security

```
docker login private-registry.io
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to
https://hub.docker.com to create one.
Username: registry-user
Password:
WARNING! Your password will be stored unencrypted in /home/vagrant/.docker/config.json.

Login Succeeded
```

```
docker run private-registry.io/apps/internal-app
```

Whitelist Allowed Registries



Whitelist Allowed Registries

Admission Configuration

<path-to-kubeconfig-file>

```
clusters:
- name: name-of-remote-imagepolicy-service
  cluster:
    certificate-authority: /path/to/ca.pem
    server: https://images.example.com/policy

users:
- name: name-of-api-server
  user:
    client-certificate: /path/to/cert.pem
    client-key: /path/to/key.pem
```

/etc/kubernetes/admission-config.yaml

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

Whitelist Allowed Registries

Enable Admission Controllers

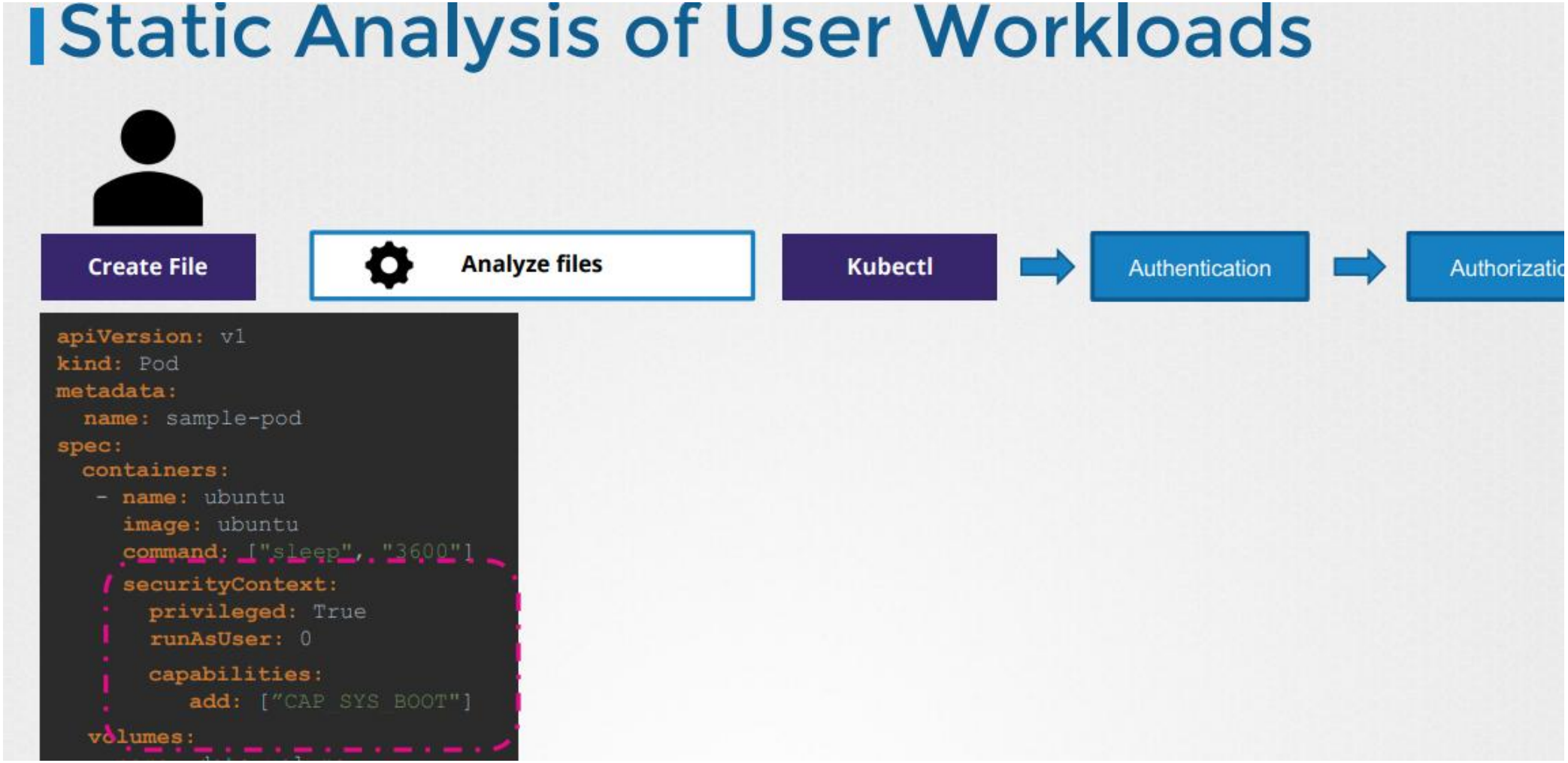
kube-apiserver.service

```
ExecStart=/usr/local/bin/kube-apiserver \\
  --advertise-address=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --authorization-mode=Node,RBAC \\
  --bind-address=0.0.0.0 \\
  --enable-swagger-ui=true \\
  --etcd-servers=https://127.0.0.1:2379 \\
  --event-ttl=1h \\
  --runtime-config=api/all \\
  --service-cluster-ip-range=10.32.0.0/24 \\
  --service-node-port-range=30000-32767 \\
  --v=2
  --enable-admission-plugins=ImagePolicyWebhook
  --admission-control-config-file=/etc/kubernetes/admission-config.yaml
```

/etc/kubernetes/manifests/kube-apiserver.yaml

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --authorization-mode=Node,RBAC
    - --advertise-address=172.17.0.107
    - --allow-privileged=true
    - --enable-bootstrap-token-auth=true
    - --enable-admission-plugins=ImagePolicyWebhook
    - --admission-control-config-file=/etc/kubernetes/admission-c
    image: k8s.gcr.io/kube-apiserver-amd64:v1.11.3
    name: kube-apiserver
```

Static Analysis of Workloads



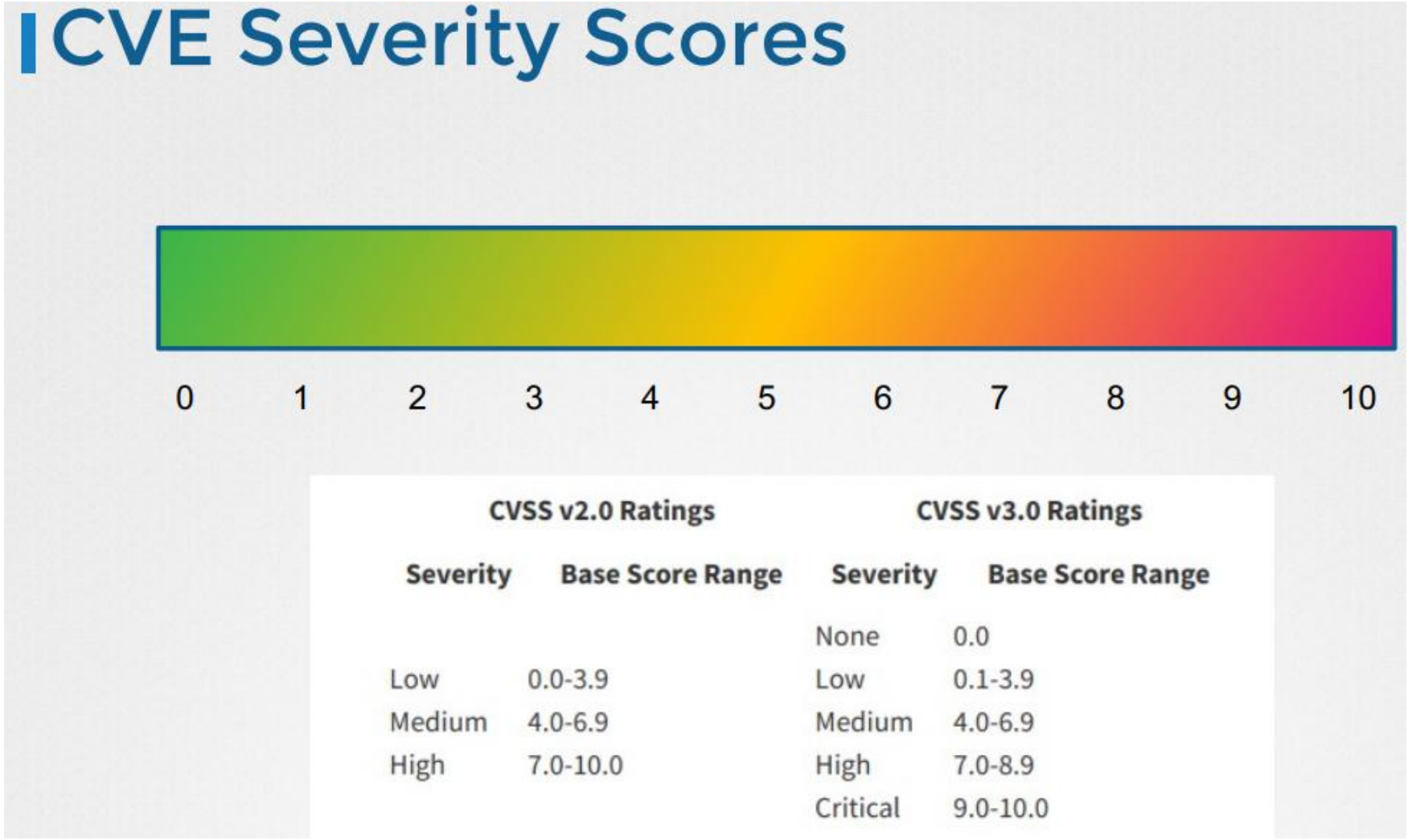
Static Analysis of Workloads

kubesec

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
spec:
  containers:
   - name: ubuntu
     image: ubuntu
     command: ["sleep", "3600"]
     securityContext:
       privileged: True
       runAsUser: 0
       capabilities:
          add: ["CAP_SYS_BOOT"]
  volumes:
  - name: data-volume
    hostPath:
       path: /data
       type: Directory
```

https://kubesec.io/

Scanning Images for CVE



Scanning Images for CVE

Trivy

```
trivy image --severity CRITICAL nginx:1.18.0
```

```
trivy image --severity CRITICAL,HIGH nginx:1.18.0
```

```
trivy image --ignore-unfixed nginx:1.18.0
```

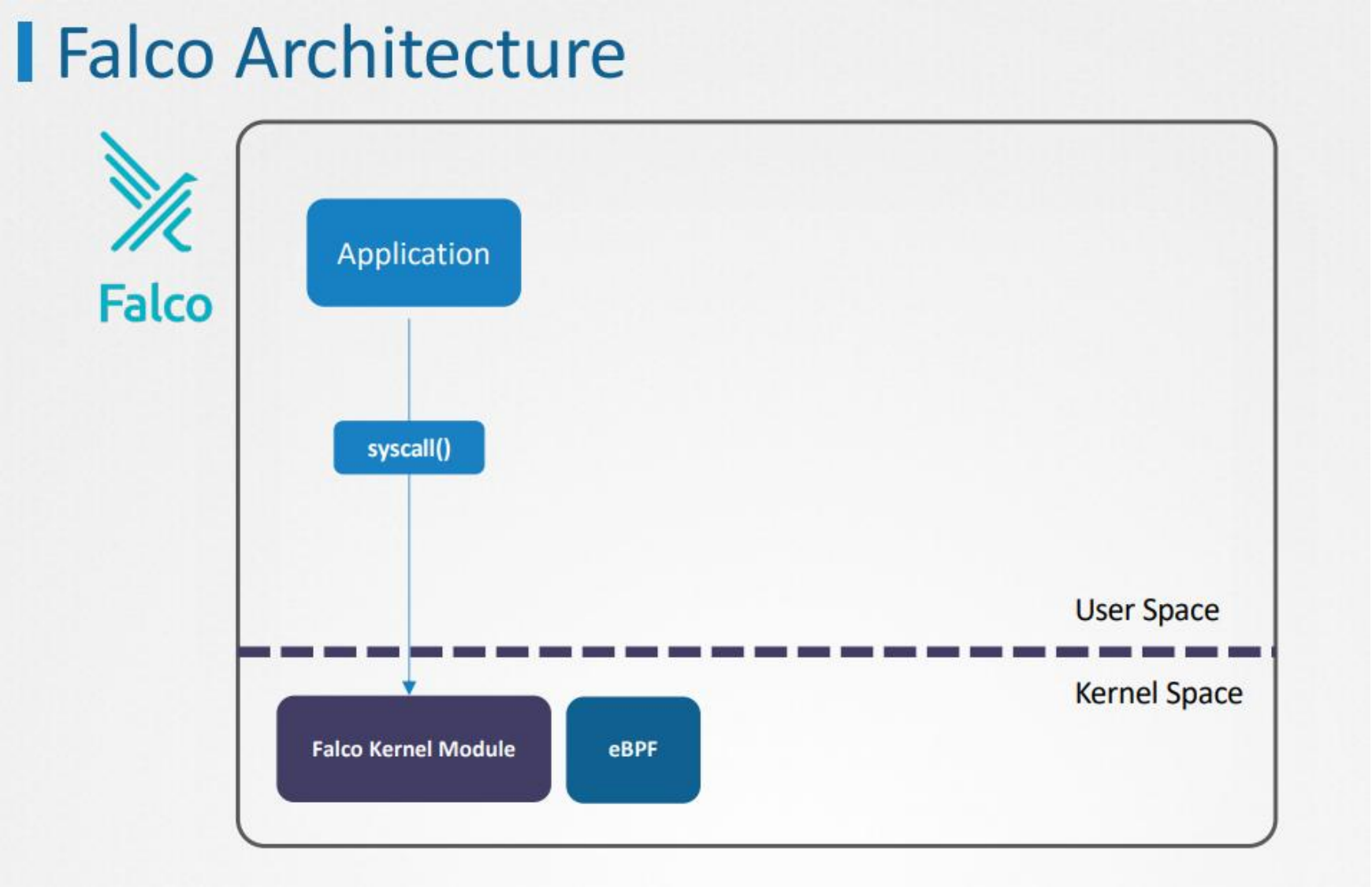
```
docker save nginx:1.18.0 > nginx.tar
```

```
trivy image --input archive.tar
```

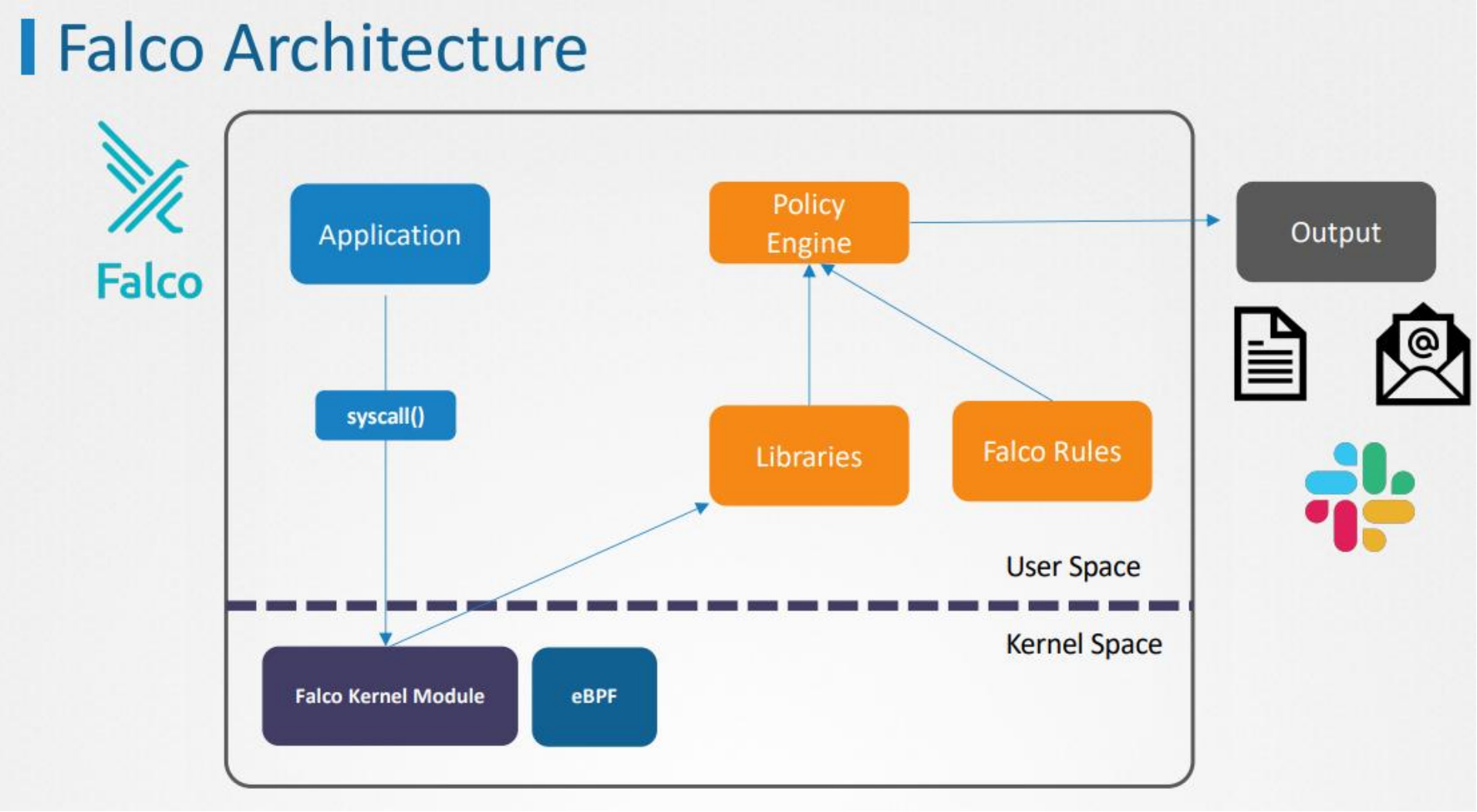
Scanning Images for CVE

- Continuously rescan images
- Kubernetes Admission Controllers to scan images
- Have your own repository with pre-scanned images ready to go
- Integrate scanning into your CI/CD pipeline

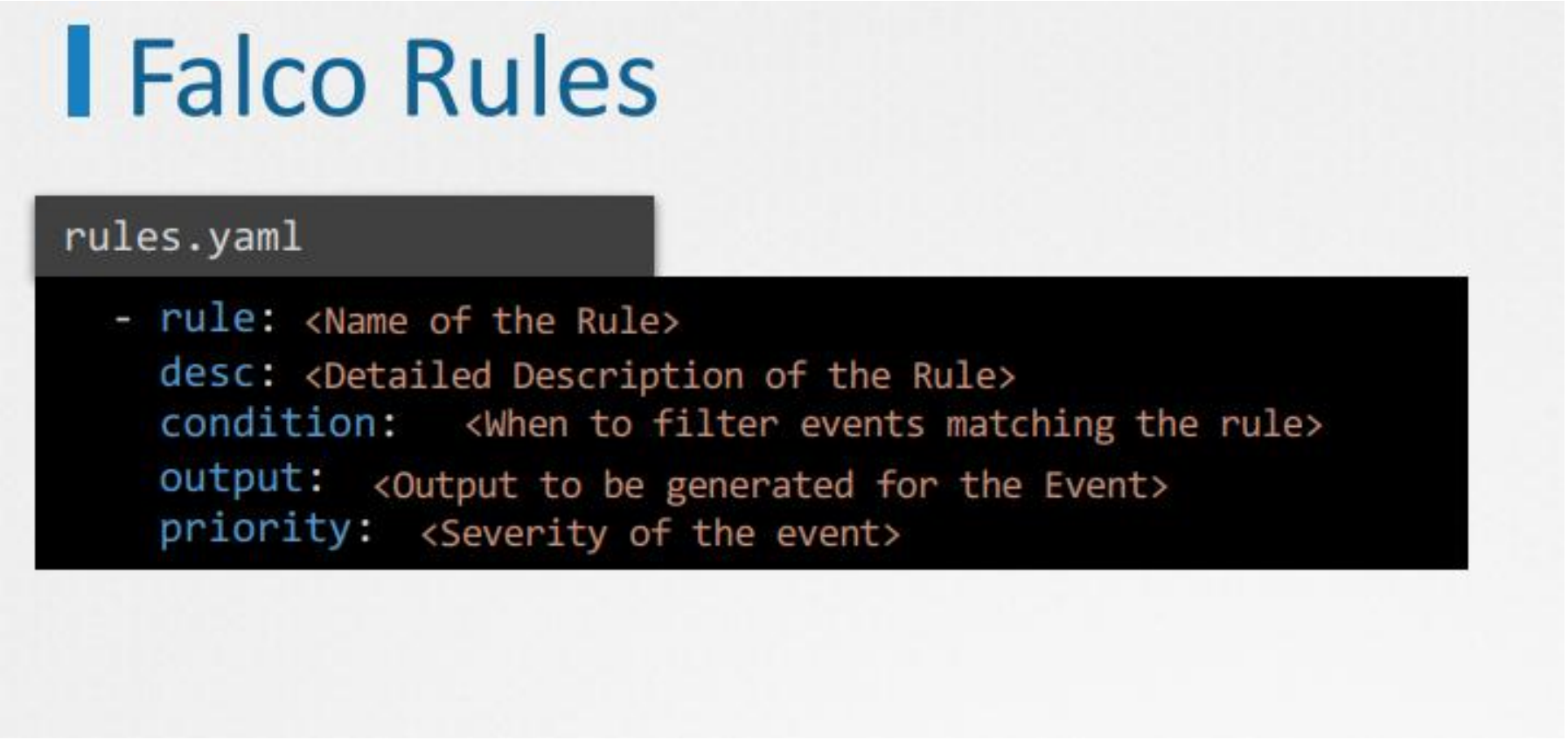
Analysis of syscalls: Falco



Analysis of syscalls: Falco



Falco Rules


Falco Rules
rules.yaml
- rule: <Name of the Rule>
desc: <Detailed Description of the Rule>
condition: <When to filter events matching the rule>
output: <Output to be generated for the Event>
priority: <Severity of the event>

Falco Rules

rules.yaml

```
- rule: Detect Shell inside a container
  desc: Alert if a shell such as bash is open inside the container
  condition: container.id != host and proc.name = bash
  output: Bash Opened (user=%user.name container=%container.id)
  priority: WARNING
```

container.id

proc.name

fd.name

evt.type

user.name

container.image.repository

DEBUG

INFORMATIONAL

NOTICE

WARNING

ERROR

CRITICAL

ALERT

EMERGENCY

Falco Configuration Files

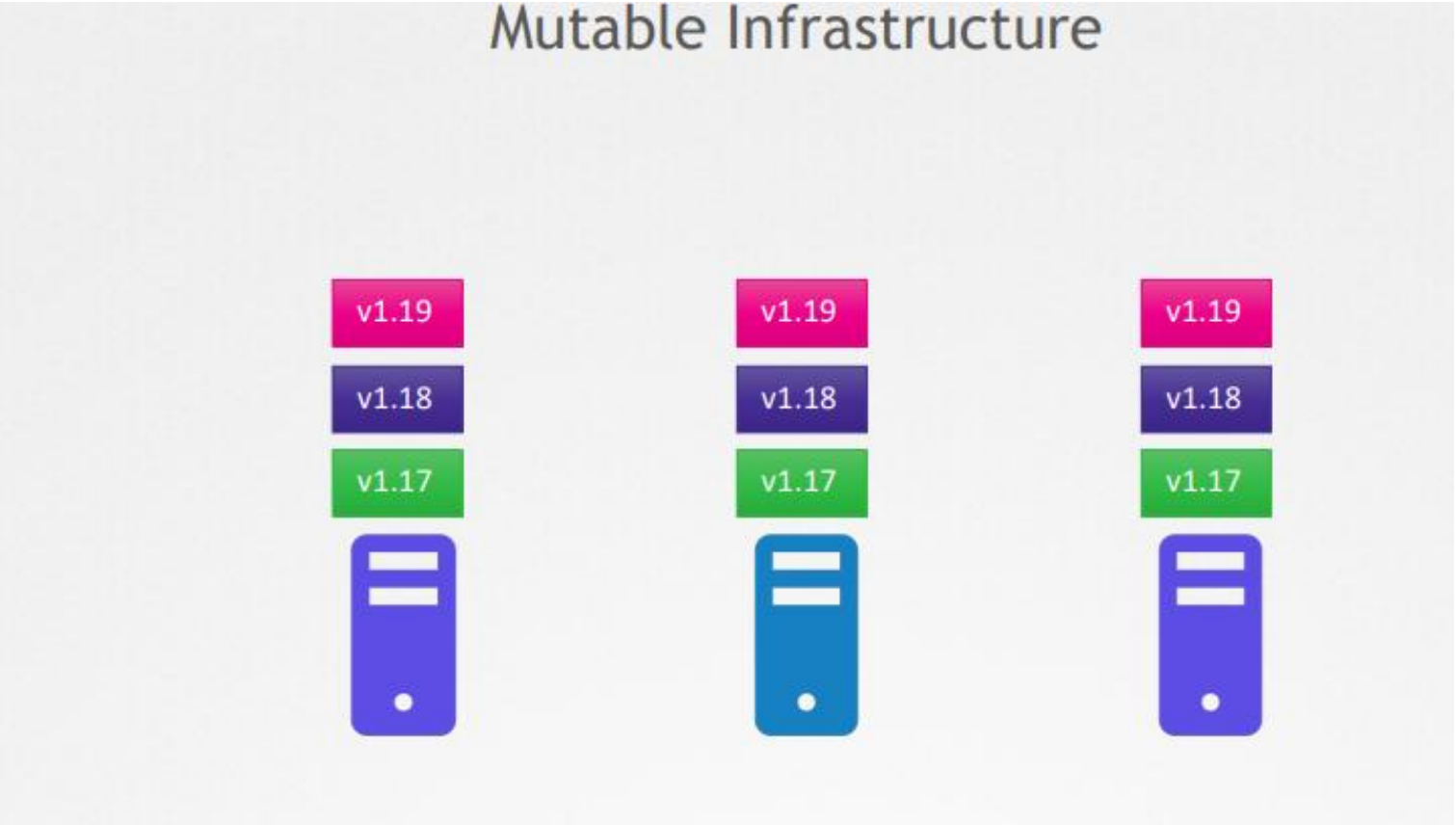
```
.
rules_file:
  - /etc/falco/falco_rules.yaml
  - /etc/falco/falco_rules.local.yaml
  - /etc/falco/k8s_audit_rules.yaml
  - /etc/falco/rules.d


json_output: false

log_stderr: true
log_syslog: true
log_level: info

priority:  debug
```

Falco Configuration Files

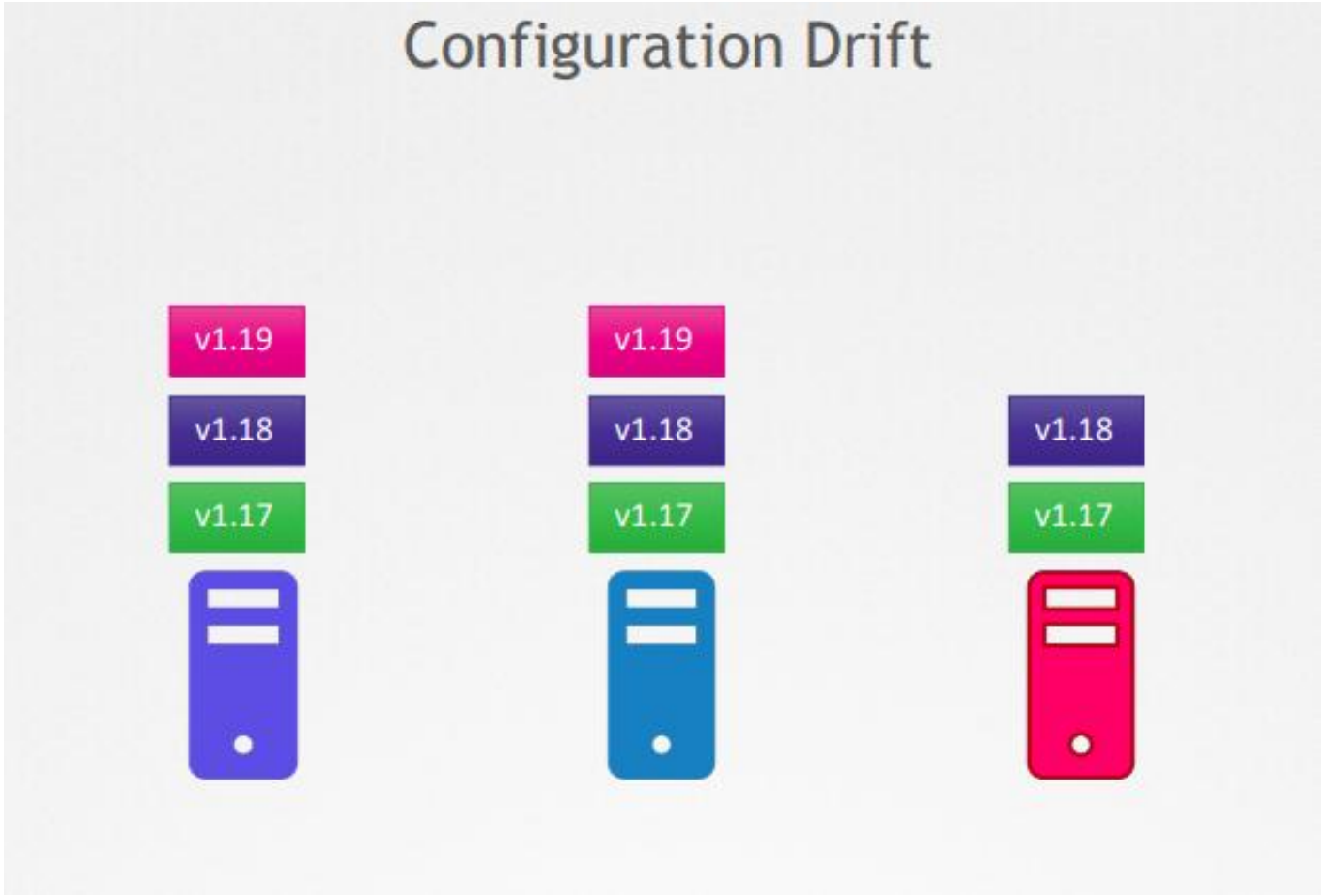
/etc/falco/falco.yaml

```
stdout_output:
  enabled: true

file_output:
  enabled: true
  filename: /opt/falco/events.txt

program_output:
  enabled: true
  program: "jq '{text: .output}' | curl -d @- -X POST https://hooks.slack.com/services/XXX"

http_output:
  enabled: true
  url: http://some.url/some/path/
```

Immutable Infrastructure



Immutable Infrastructure



Immutable Infrastructure

```
kubectl create -f nginx.yaml
pod/nginx created
```

```
kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Error     0          20s
```

nginx.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    securityContext:
          readOnlyRootFilesystem: true
```

Immutable Infrastructure

```
kubectl create -f nginx.yaml
pod/nginx created
```

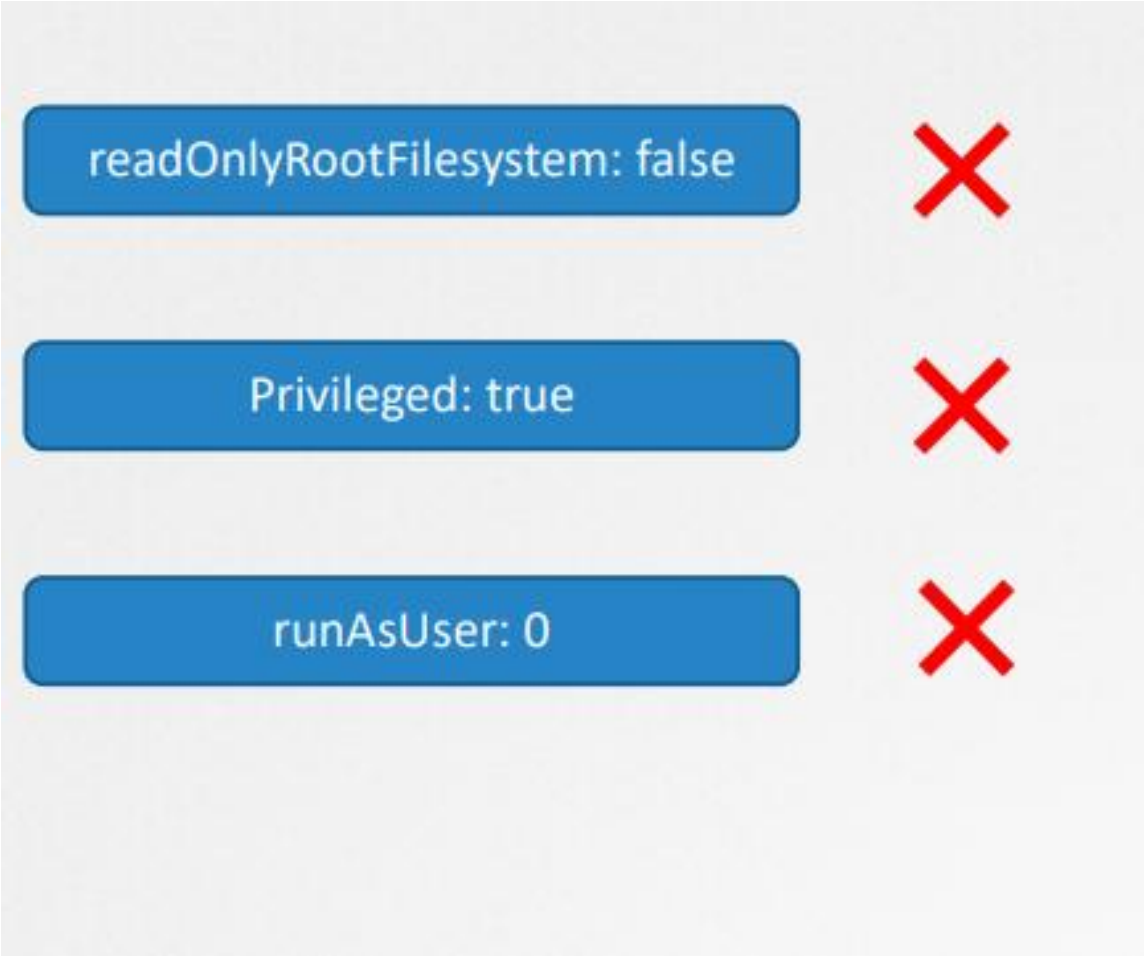
```
kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          20s
```

```
kubectl exec -ti nginx -- apt update
Reading package lists... Done
E: List directory /var/lib/apt/lists/partial is missing. - Acquire
(30: Read-only file system)
command terminated with exit code 100
```

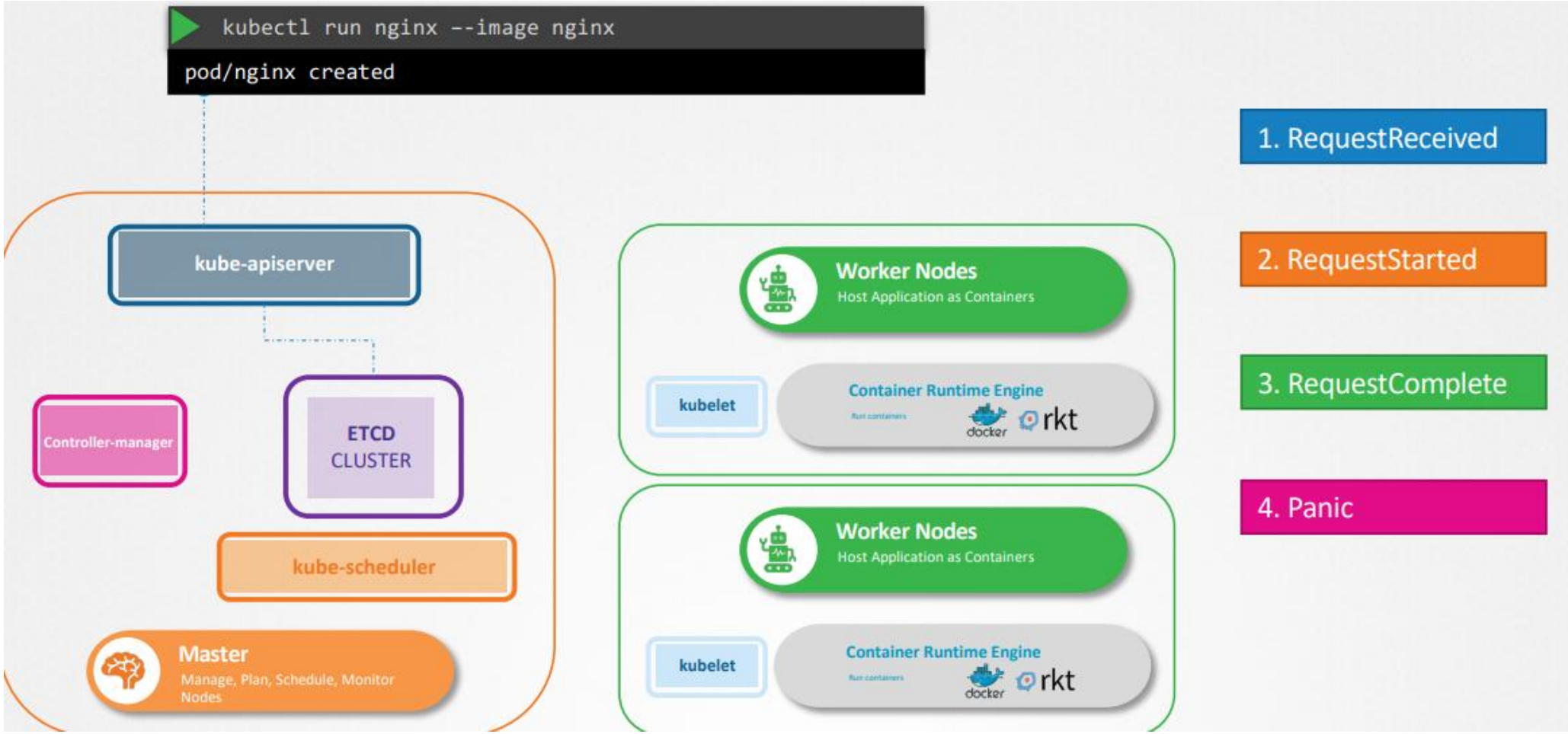
nginx.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx

    securityContext:
      readOnlyRootFilesystem: true
      privileged: true
    volumeMounts:
      - name: cache-volume
        mountPath: /var/cache/nginx
      - name: runtime-volume
        mountPath: /var/run
  volumes:
  - name: cache-volume
    emptyDir: {}
  - name: runtime-volume
    emptyDir: {}
```

Immutable Infrastructure



Kubernetes Auditing



Kubernetes Auditing

audit-policy.yaml

```yaml
apiVersion: audit.k8s.io/v1
kind: Policy
omitStages: ["RequestReceived"]
rules:
  - namespace: ["prod-namespace"]
    verb: ["delete"]
    resources:
    - groups: " "
      resources: ["pods"]
      resourceNames: ["webapp-pod"]
    level:  RequestResponse

  - level: Metadata
    resources:
    - groups: " "
      resources: ["secrets"]
```

Kubernetes Auditing

/etc/kubernetes/manifests/kube-apiserver.yaml

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --authorization-mode=Node,RBAC
    - --advertise-address=172.17.0.107
    - --allow-privileged=true
    - --enable-bootstrap-token-auth=true
    - --audit-log-path=/var/log/k8-audit.log
    - --audit-policy-file=/etc/kubernetes/audit-policy.yaml
```