KUBERNETES CONFIGURATIONS

Author: Mehul Solanki

Date: 13-July-2024

Version: 1.0

Document History

Author	*Date*	*Comments*
Mehul Solanki	13-July-2024	

Contact Details

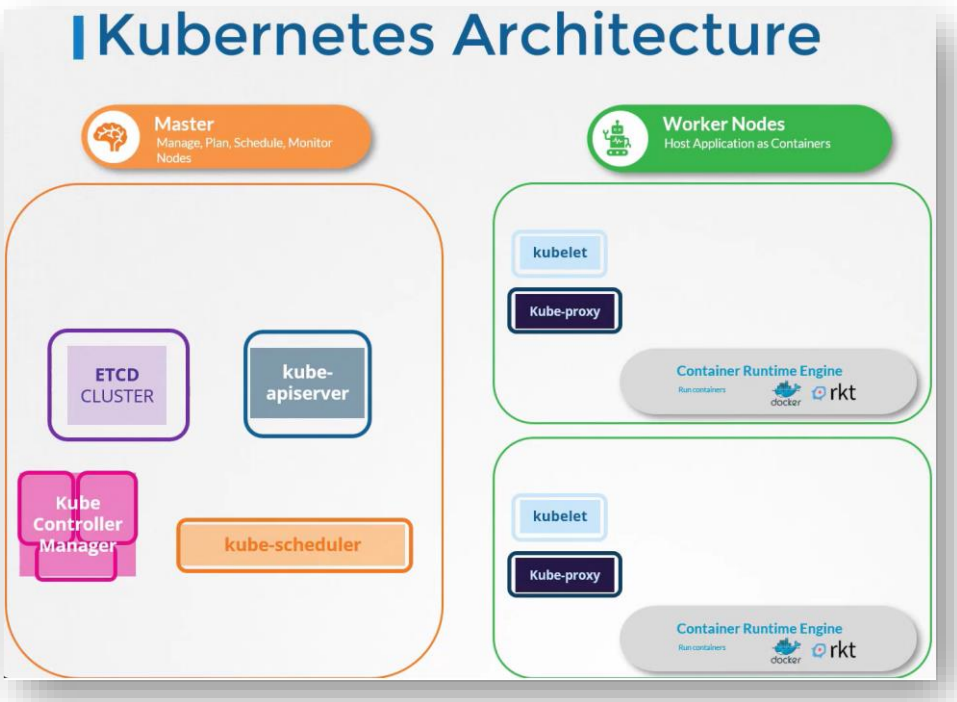
Name : Mehul Solanki
Phone : +919879729795
Email : er.mehulsolanki1887@gmail.com
LinkedIn : https://in.linkedin.com/in/mehulsolanki1887

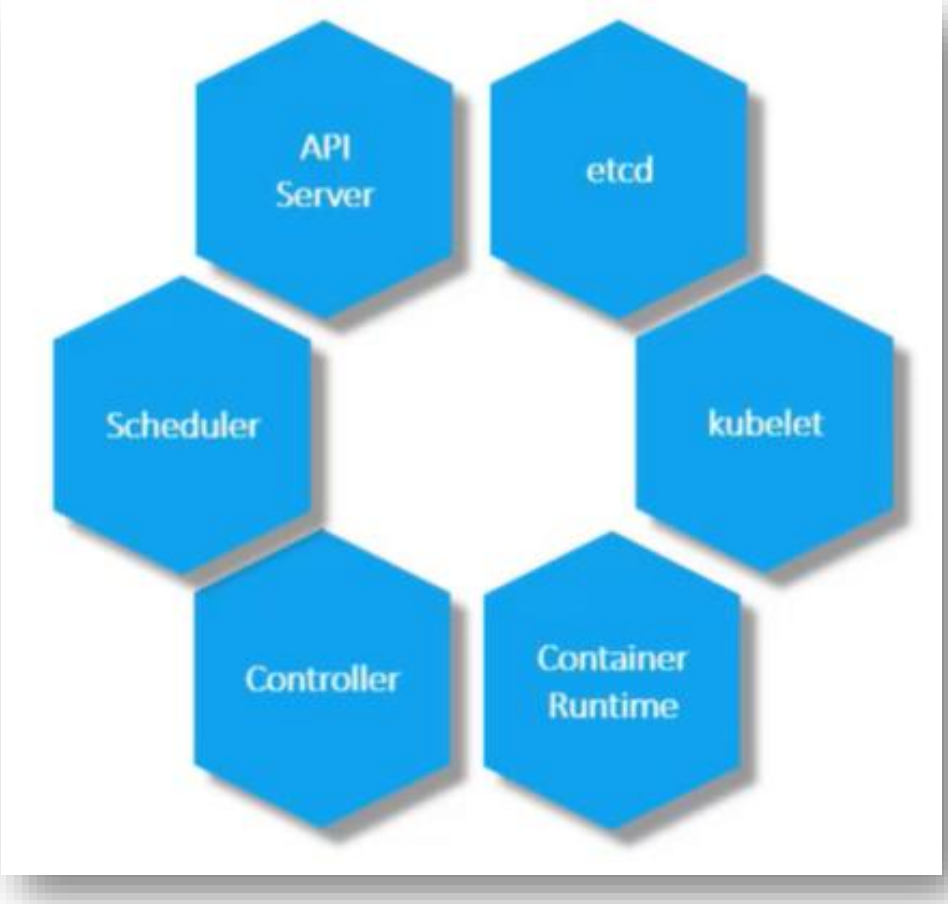
Table of Contents

Table of Figures

1. CONTENTS

2.1 CLUSTER ACHITECTURE





2.1.1 MASTERNODES (Manage, Plan, Schedule, Monitor, Nodes)

2.1.1.1 Kubeadm (Provisioner)

2.1.1.2 ETED Cluster (Database)

2.1.1.3 Kube-Scheduler (Scheduler)

2.1.1.4 Controller-Manager (Controller)

- Node-Controller (Check to Onboarding new nodes container to the cluster)
- Replications Controller (Check to Desire containers are running at all times)

2.1.1.5 Kube-APIServer (A primary management component to responsible for orchestrating all operations within the cluster)

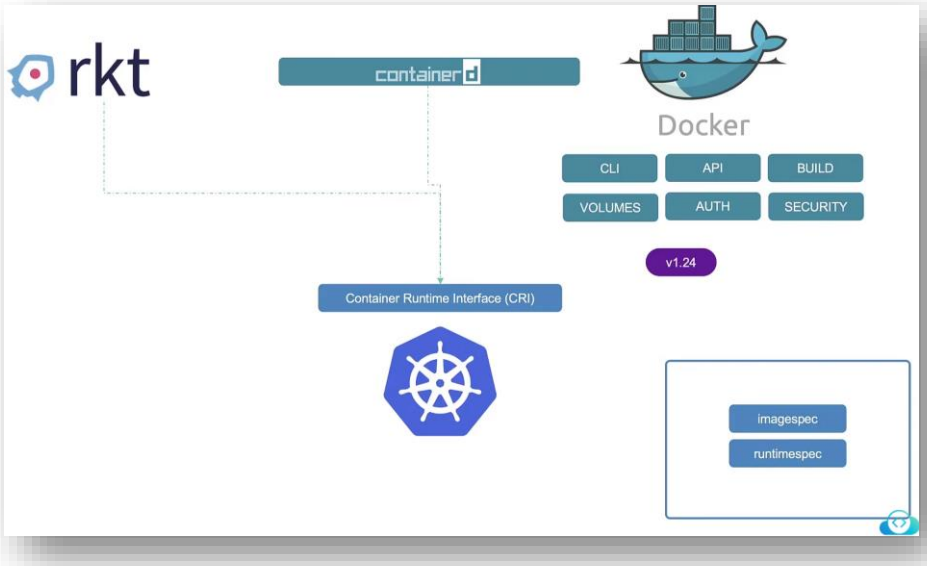
2.1.1.6 Container Runtime Engine (ContainerD, Rocket)

2.1.2 WORKER NODES (Host Applications as Container)

- Kubelet (It is an agent listen the instructions from apiserver to deploy and destroy containers)
- Kubeproxy (It's help to communicate between two containers in different nodes)

2.2 DOCKER-VS-CONTAINER-D

2.2.1

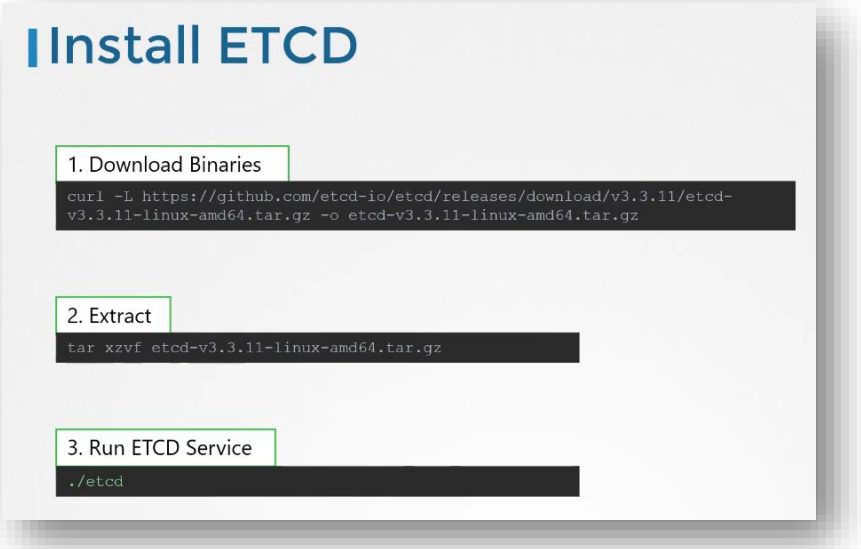


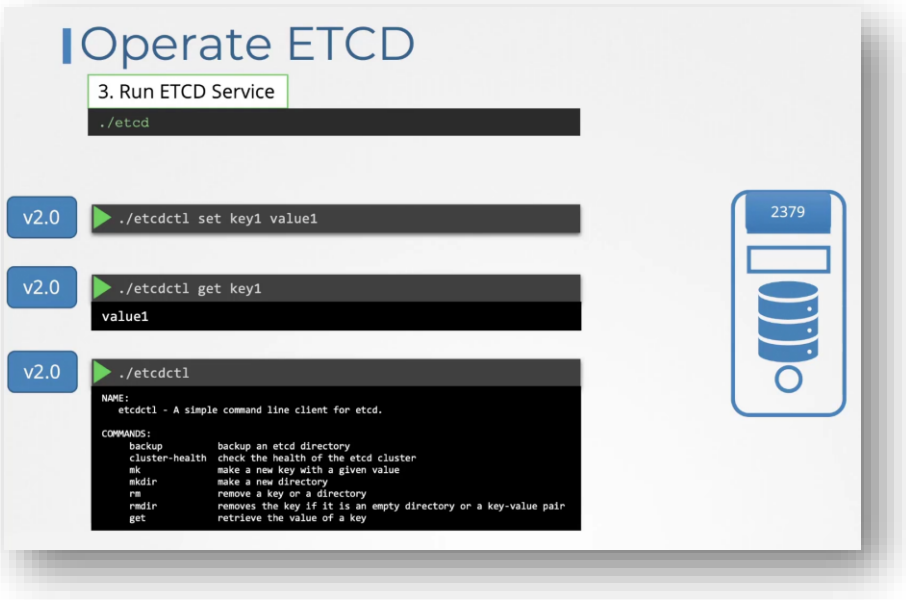
2.2.2

Retrieve debugging information

docker cli	crictl	Description	Unsupported Features
attach	attach	Attach to a running container	--detach-keys , --sig-proxy
exec	exec	Run a command in a running container	--privileged , --user , --detach-keys
images	images	List images	
info	info	Display system-wide information	
inspect	inspect , inspecti	Return low-level information on a container, image or task	
logs	logs	Fetch the logs of a container	--details
ps	ps	List containers	
stats	stats	Display a live stream of container(s) resource usage statistics	Column: NET/BLOCK I/O, PIDs
version	version	Show the runtime (Docker, ContainerD, or others) version information	

2.3 ETCD



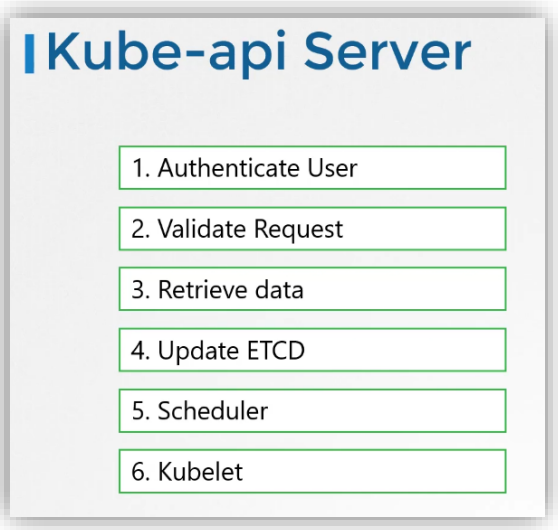


2.3.1 ETCD-COMMANDS

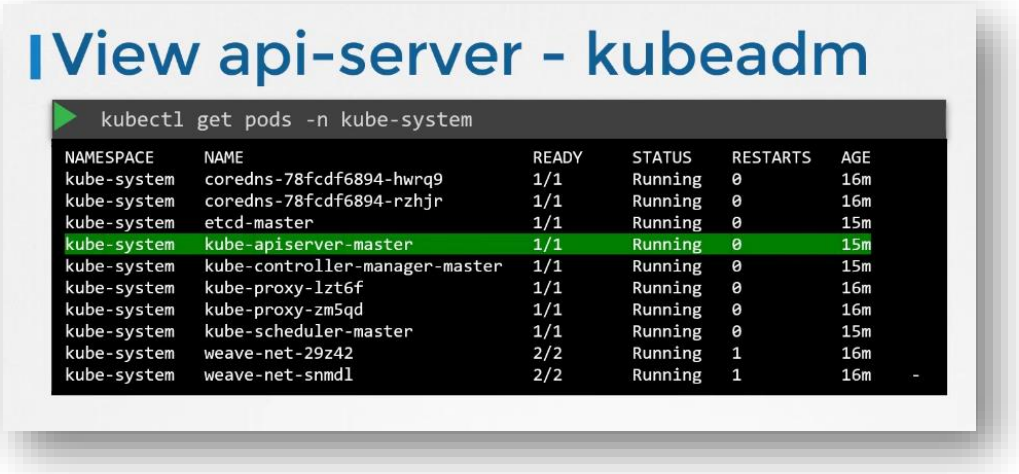
- etcdctl backup
- etcdctl cluster-health
- etcdctl mk
- etcdctl mkdir
- etcdctl set
- etcdctl snapshot save
- etcdctl endpoint health
- etcdctl get
- etcdctl put
- export ETCDCTL_API=3
- --cacert /etc/kubernetes/pki/etcd/ca.crt
- --cert /etc/kubernetes/pki/etcd/server.crt
- --key /etc/kubernetes/pki/etcd/server.key
- kubectl exec etcd-controlplane -n kube-system -- sh -c "ETCDCTL_API=3 etcdctl get / --prefix --keys-only --limit=10 --cacert /etc/kubernetes/pki/etcd/ca.crt --cert /etc/kubernetes/pki/etcd/server.crt --key /etc/kubernetes/pki/etcd/server.key"

2.4 KUBE-API SERVER

2.4.1



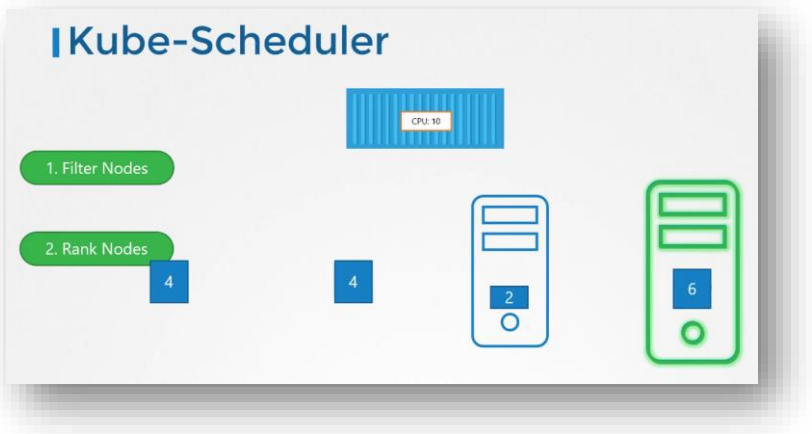
2.4.2



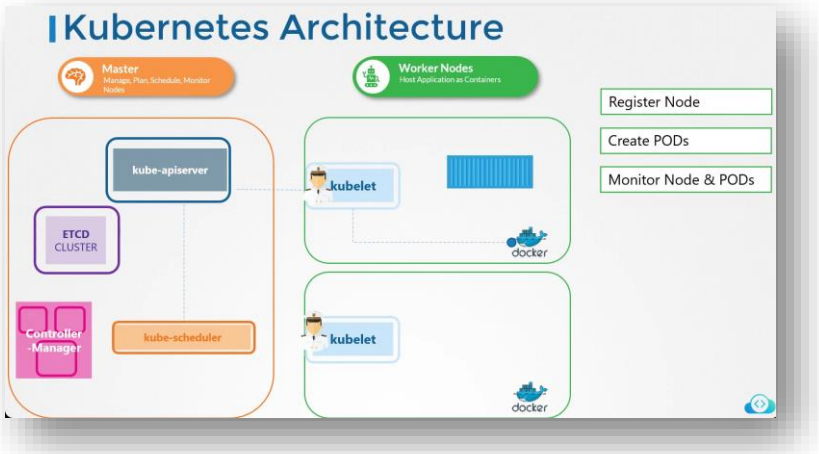
2.5 KUBECONTROL MANAGER

- Node Controller
- Replications Controller

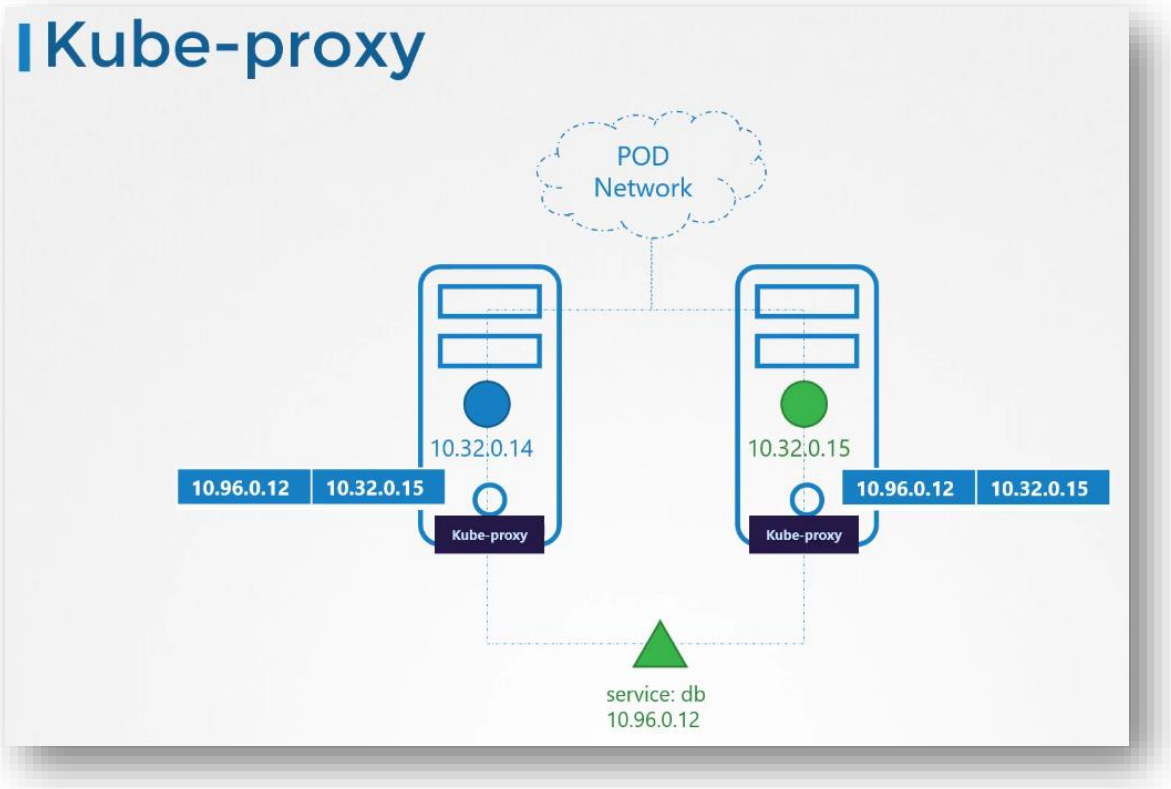
2.6 KUBE SCHEDULER

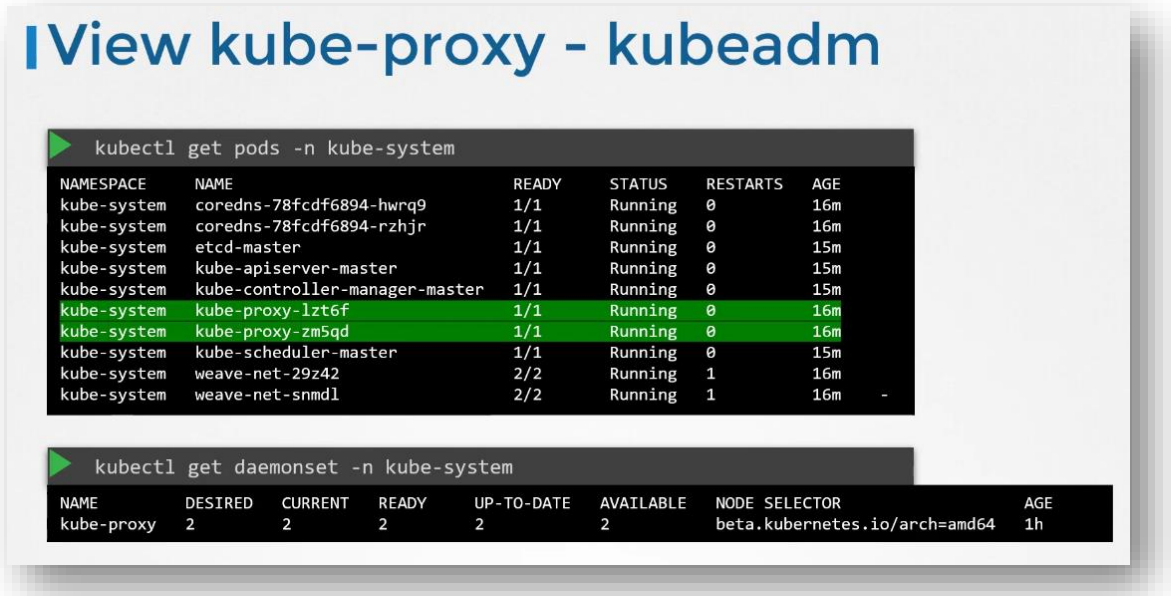


2.7 KUBELET



2.8 KUBEPROXY



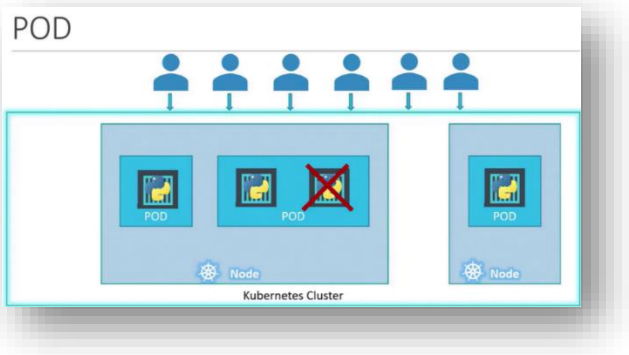


2.9 PODS

2.9.1 COMPONENETS

- Containers (Single and Multi)
- IP Address
- Shared Storage
- Life Cycle
- Create
- Run
- Terminate

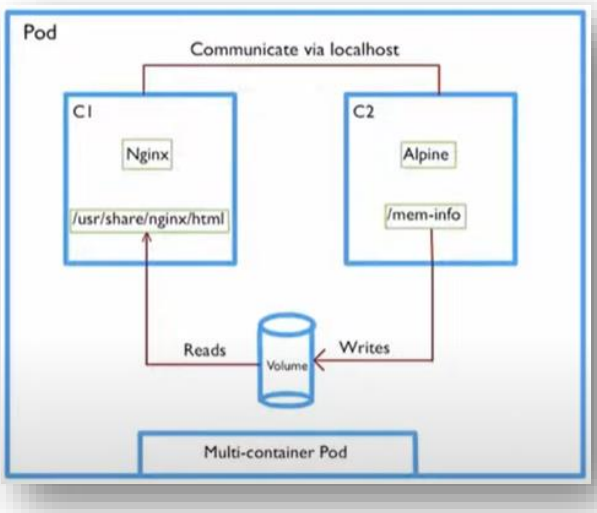
2.9.2 PODS STRUCTURE



2.9.3 COMMANDLINE

- kubectl run my-pod --image=nginx
- kubectl get my-pod
- kubectl describe pod my-pod
- kubectl get pods -o wide
- kubectl get pods -o wide --show-labels
- kubectl exec -it my-pod -- bash
- kubectl run my-pod --image=redis --image=nginx

2.10 MULTI CONTAINER IN A SINGLE POD

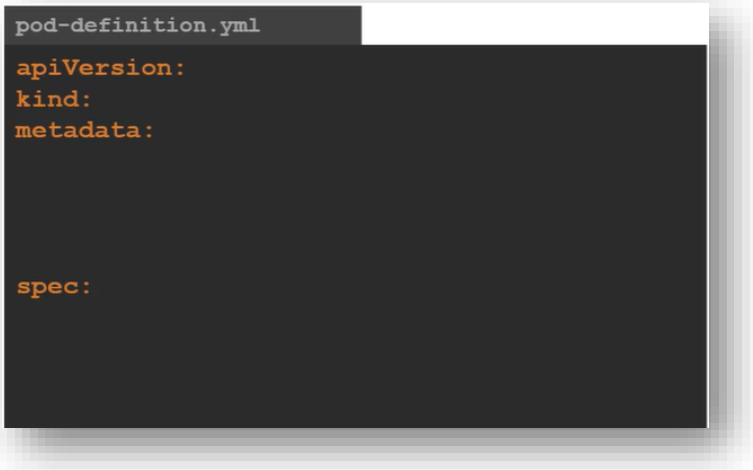



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: redis-django
  labels:
    app: web
spec:
  containers:
    - name: key-value-store
      image: redis
      ports:
        - containerPort: 6379
    - name: frontend
      image: django
      ports:
        - containerPort: 8000
```

2.10.1 COMMANDLINE

- kubectl exec -it my-pod -c container1 -- bash

2.11 POD YML STRUCTURE

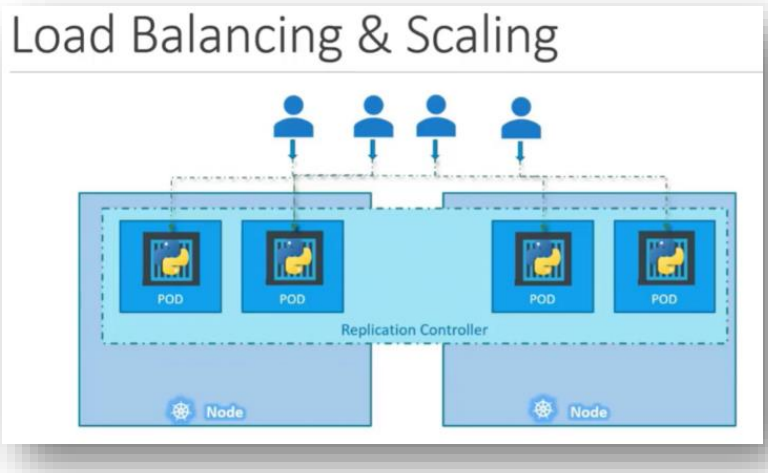


2.11.1 COMMAND LINE

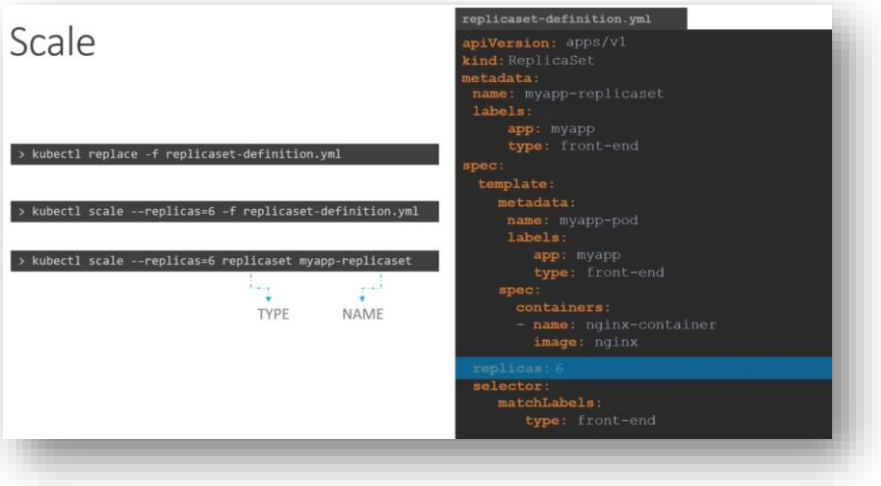
- kubectl run myapp-pod --image=nginx --dry-run=client -o yaml >> pod-definition.yaml
- kubectl create -f prod-definition.yaml

2.11.2 BASIC YML FILE

2.12 REPLICASET



2.12.1



2.12.2 COMMANDLINE

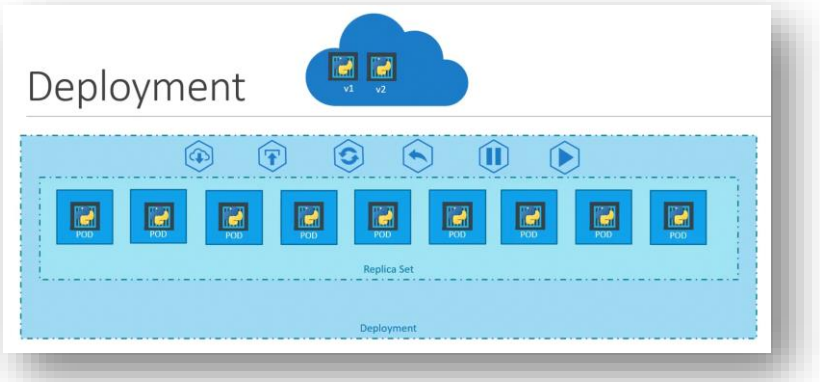
- kubectl api-resources | grep replicaset (To Check api Version)
- kubectl explain replicaset | grep VERSION (To Check replicaset version).
- kubectl create -f rc-definition.yml
- kubectl get replicaset
- kubectl describe replicaset myapp-replicaset
- kubectl replace -f rc-definition.yml
- kubectl edit replicasets myapp-replicaset
- kubectl delete replicaset myapp-replicaset
- kubectl scale replicaset myapp-replicaset --replicas=5
- kubectl scale replicas=6 -f rc-definition.yml
- BASIC YML (REPLICA SET)

2.13 DEPLOYMENTS

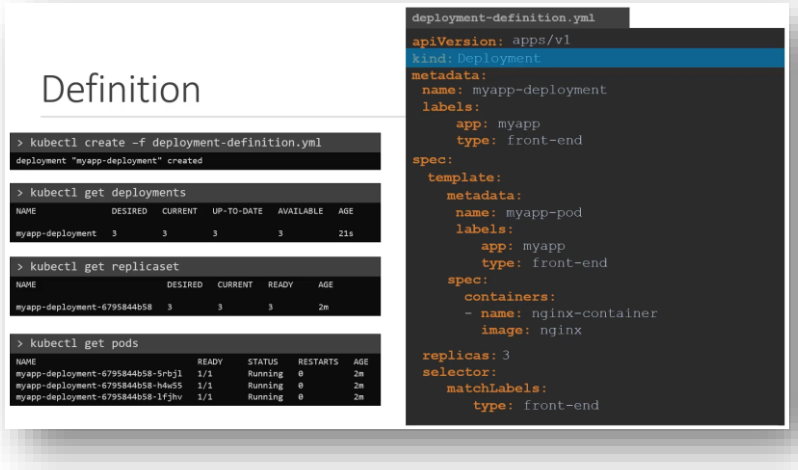
2.13.1 COMPONENETS

- Deploy
- Update
- Rollback

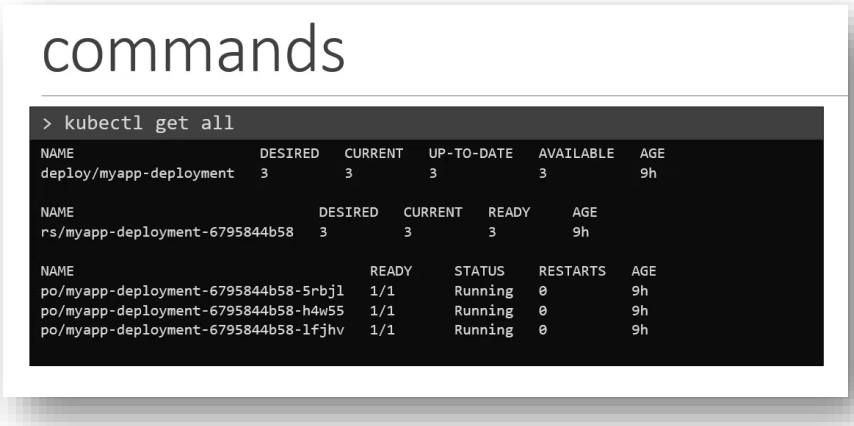
2.13.2



2.13.3



2.13.4



2.13.5 COMMAND LINE

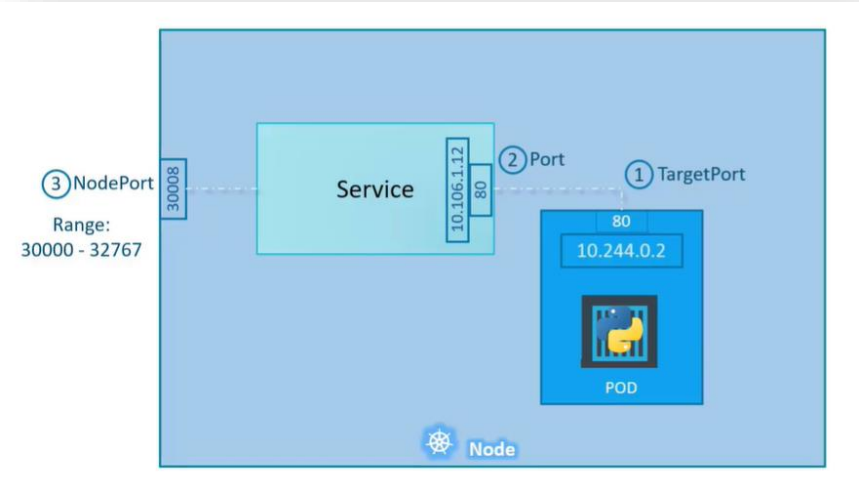
- kubectl create deployment my-dep --image=nginx
- kubectl create -f deployment-deninition.yaml
- kubectl get deployment
- kubectl describe deployment my-dep
- kubectl get replicaset
- kubectl get pods

- kubectl create deployment my-dep --image=nginx --replicas=3
- kubectl create deployment my-dep --image=nginx --replicas=3 –dry-run=client -o yaml >> deployment-definition.yaml
- kubectl get replicaset
- kubectl create -f deployment-deninition.yaml
- kubectl get deployment --watch

2.14 SERVICES

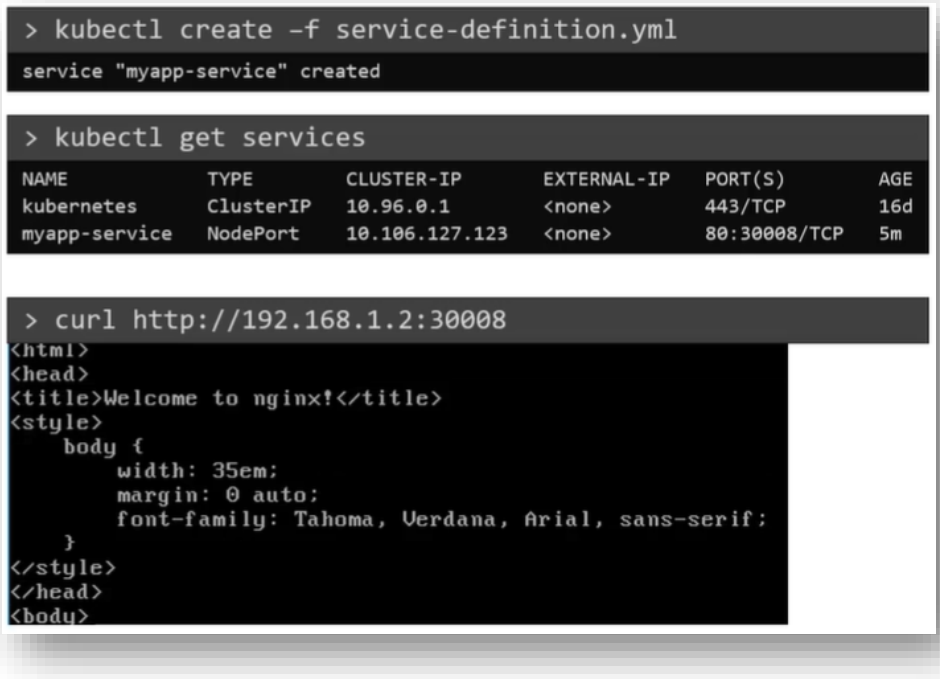
2.14.1 TYPES OF SERVICES

- NodePort



service-definition.yml

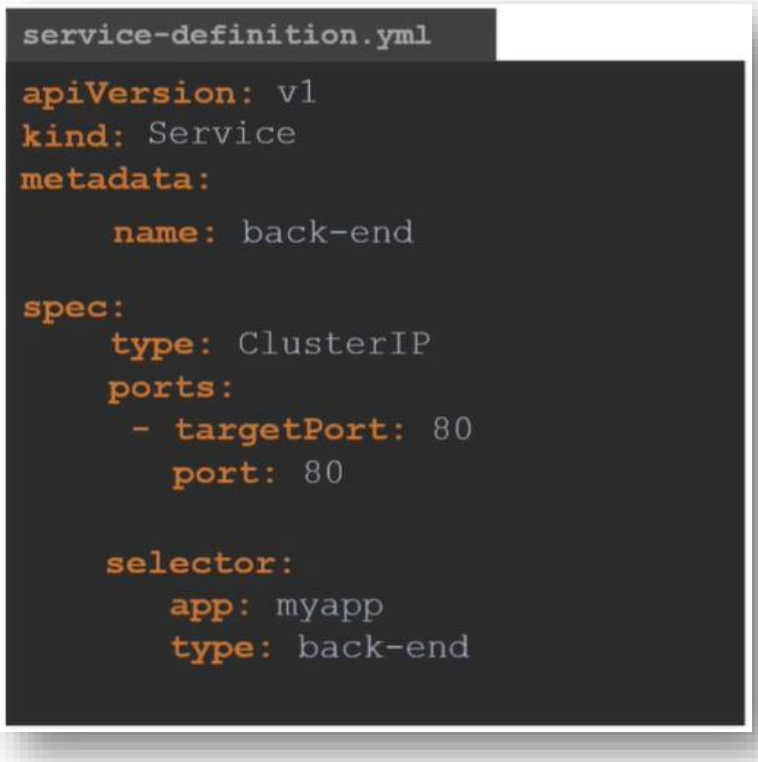
```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
     - targetPort: 80
       port: 80
       nodePort: 30008
    selector:
       app: myapp
       type: front-end
```



2.14.2 COMMANDLINE

- kubectl expose pod pod --type=NodePort --port=80 --name=my-pod
- kubectl get svc

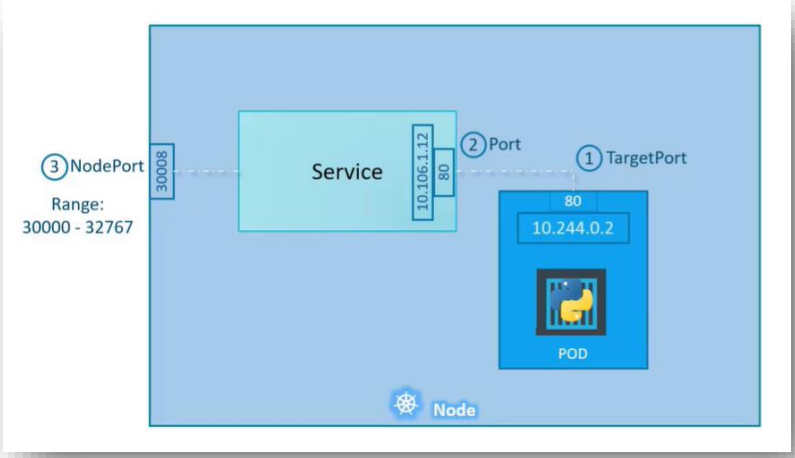
2.14.3 CLUSTER IP

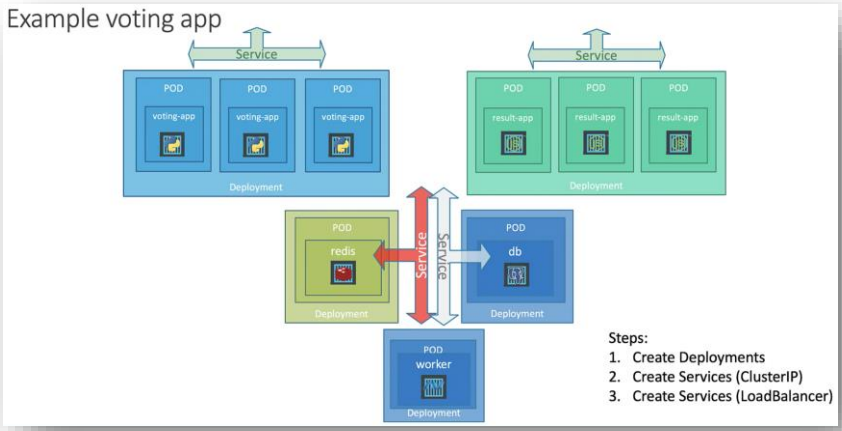


2.14.4 COMMANDLINE

- kubectl expose pod pod --type=ClusterIP --port=80 --name=my-pod
- kubectl get svc

2.14.5 LOAD BALANCER





service-definition.yml

```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: LoadBalancer
    ports:
      - targetPort: 80
        port: 80
        nodePort: 30008
```

2.14.6 COMMANDLINE

- kubectl expose pod pod --type=LoadBalancer --port=80 --protocol=TCP --target-port=80 --name=my-pod
- kubectl expose pod pod --type=LoadBalancer --port=80 --name=my-pod

2.15 NAMESPACE

2.15.1 AVAILABLE NAMESPACES BY DEFAULT

- DEFAULT
- KUBE-PUBLIC
- KUBE-SYSTEM

2.15.2 YML FILE

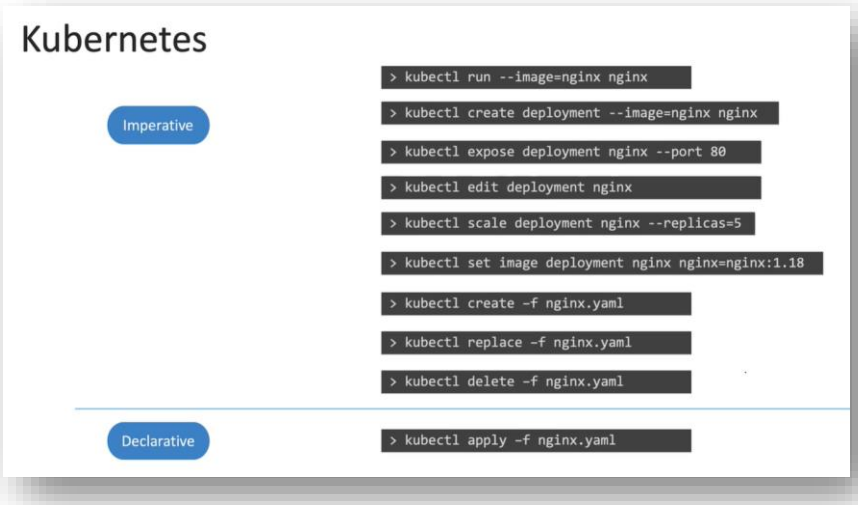
pod-definition.yml

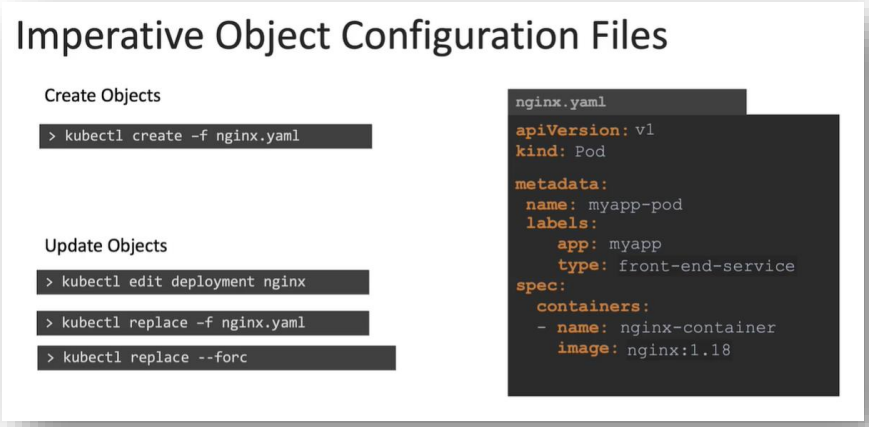
```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 namespace: dev

 labels:
    app: myapp
    type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

2.15.3 COMMADLINE

- Kubectl get namespace
- Kubectl run nginx --image=nginx -n namespace
- kubectl get pods -n namespace
- kubectl get pods --all-namespaces
- kubectl get pods --all-namespaces | grep podname
- kubectl get pods -A

2.16 IMPERATIVE V/S DECLARATIVE

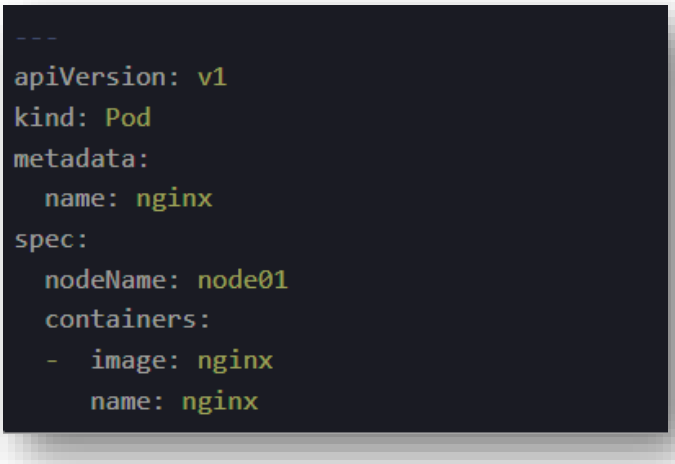




2.16.1 COMMANDLINE

3. SCHEDULING

3.1 MANUAL SCHEDULING

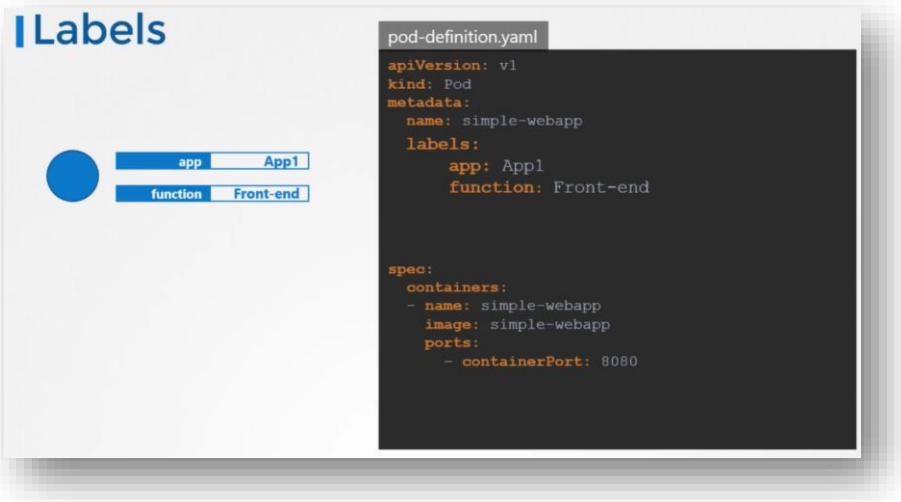


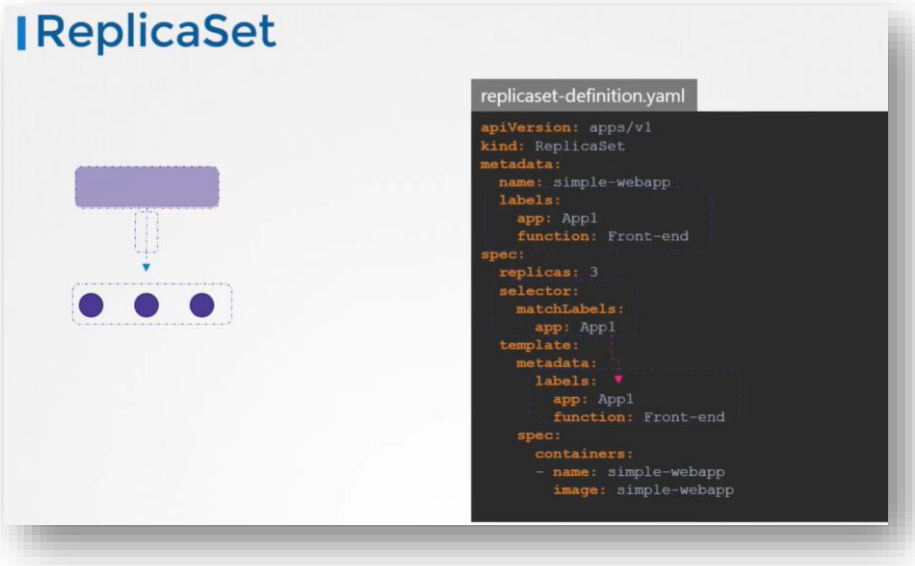
```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeName: node01
  containers:
  -  image: nginx
     name: nginx
```

3.1.1 COMMAND LINE

- Kubectl run my-pod –image=nginx
- Kubectl get pods
- kubectl get pods --namespace kube-system
- create file node1.yaml as per above
- kubectl get pods -o wide
- kubectl replace --force -f node1.yaml
- kubectl get pods --watch
- kubectl get pods -n kube-system

3.2 LABELES AND SELECTORS



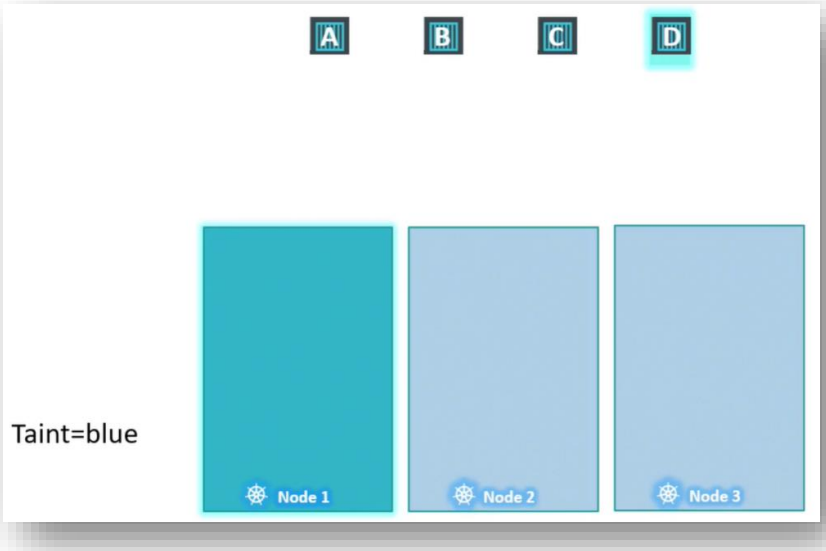


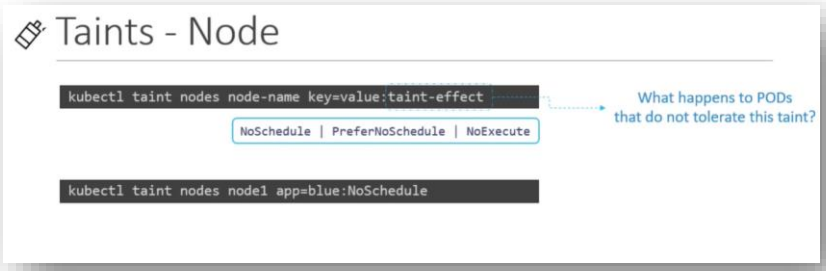
3.2.1 COMMANDLINE

- kubectl label node node01 key=value (DefaultCommanad)
- kubectl label node node01 size=large
- kubectl get selector app=my-pod
- kubectl get all --selector env=prod --no-headers | wc -l
- kubectl get all --selector env=prod,bu=finance,tier=frontend

3.3 TAINTS AND TOLERATIONS

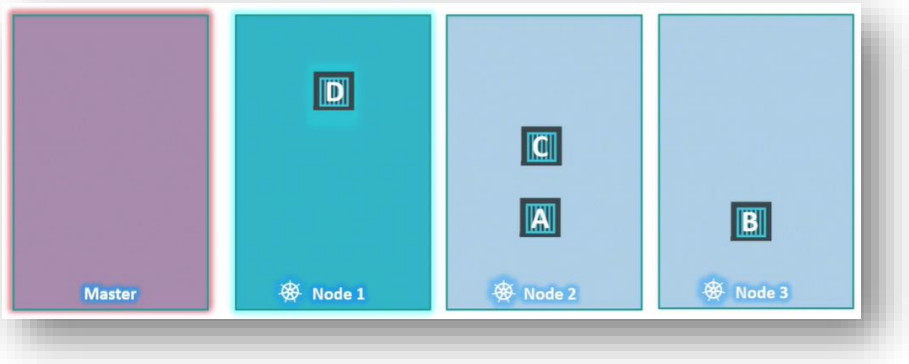
- NoSchedule
- PreferNoSchedule
- NoExecute





pod-definition.yml

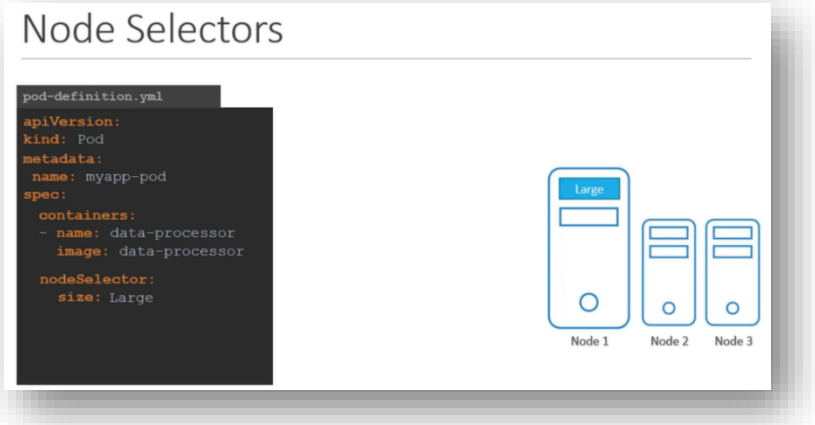
```yaml
apiVersion:
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
  - name: nginx-container
    image: nginx

  tolerations:
  - key:"app"
    operator:"Equal"
    value:" blue"
    effect:" NoSchedule"
```

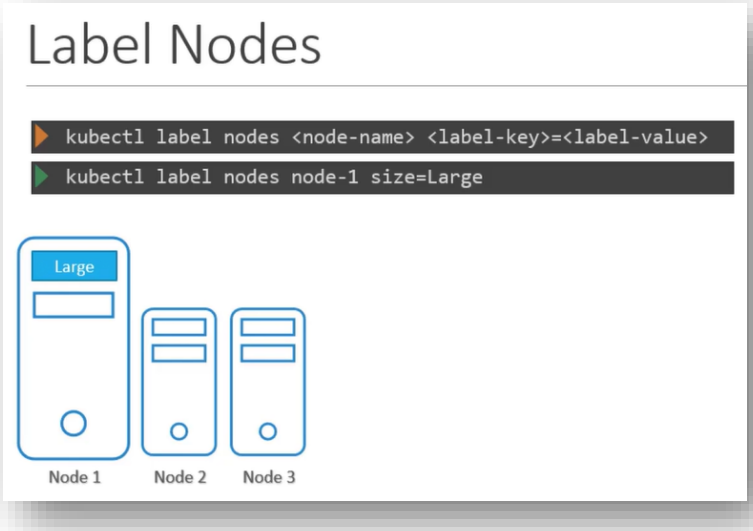


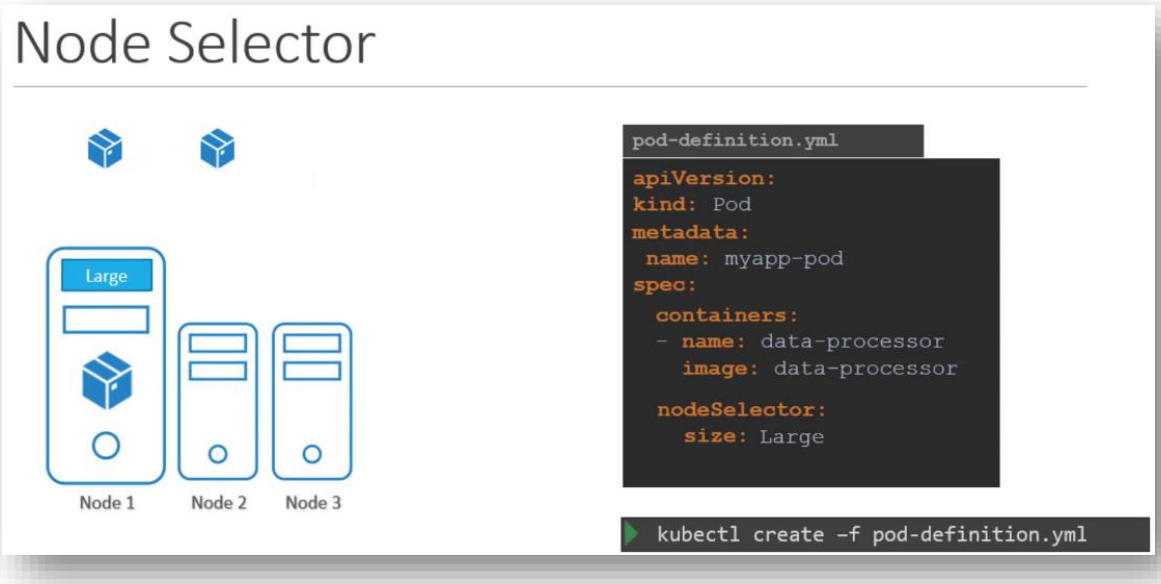
3.3.1 COMMANDLINE

- kubectl describe nodes | grep taint
- kubectl describe nodes kubemaster | grep taint
- kubectl taint nodes node01 key=value:NoSchedule (DefaultCommand)
- kubectl taint nodes node01 app=blue:NoSchedule (Example)
- kubectl taint nodes node01 spray=mortein:NoSchedule (Example)
- kubectl describe pod my-pod

3.4 NODE SELECTOR







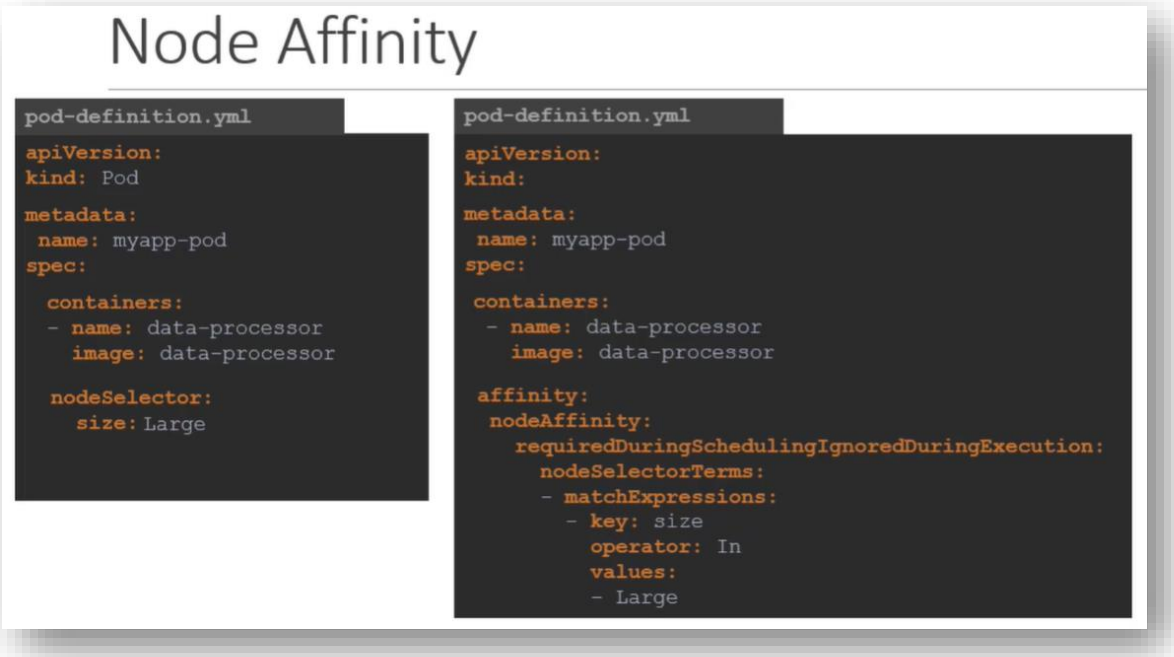
3.4.1 COMMANDLINE

- kubectl label node node01 key=value (DefaultCommanad)
- kubectl label node node01 size=large

3.5 NODE AFFINITY

3.5.1 TYPE

- **Required** DuringScheduleing **Ignore** DuringExecution
- **Preferred** DuringScheduleing **Ignore** DuringExecution
- **Required** DuringScheduleing **Required** DuringExecution
- **Preferred** DuringScheduleing **Required** DuringExecution



pod-definition.yml

```yaml
apiVersion:
kind:
metadata:
 name: myapp-pod
spec:
 containers:
  - name: data-processor
    image: data-processor
 affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: size
          operator: NotIn
          values:
          - Small
```

Node Affinity Types

Available:

requiredDuringSchedulingIgnoredDuringExecution

preferredDuringSchedulingIgnoredDuringExecution

Planned:

requiredDuringSchedulingRequiredDuringExecution

preferredDuringSchedulingRequiredDuringExecution

Node Affinity Types

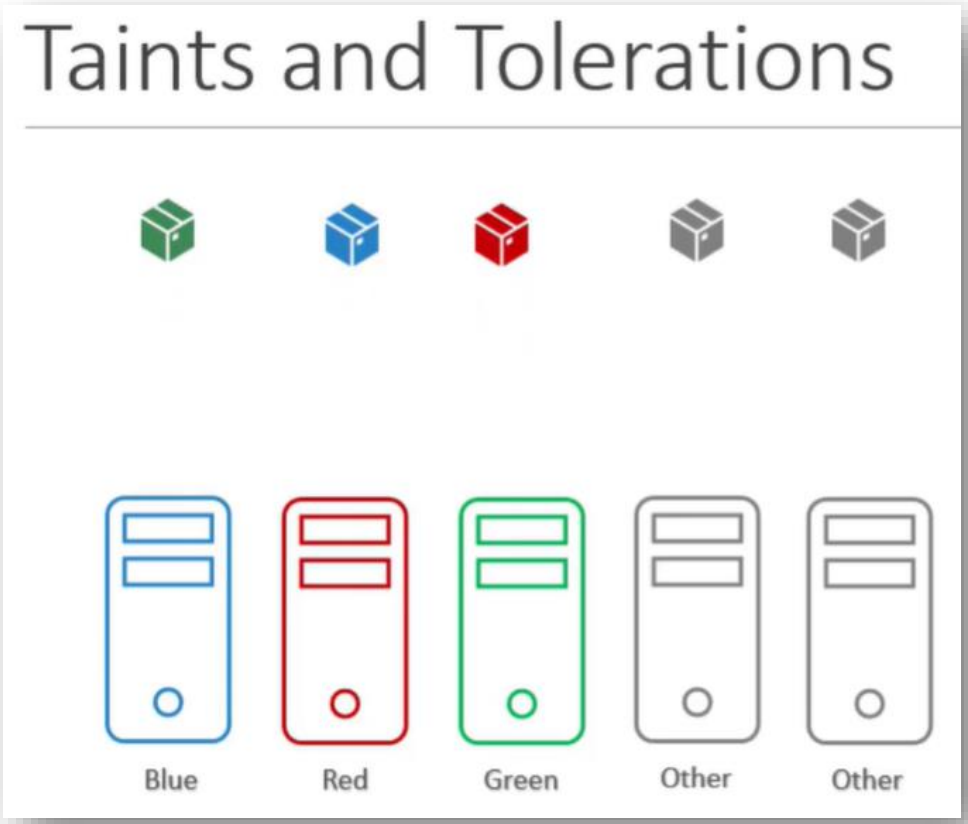
Planned:

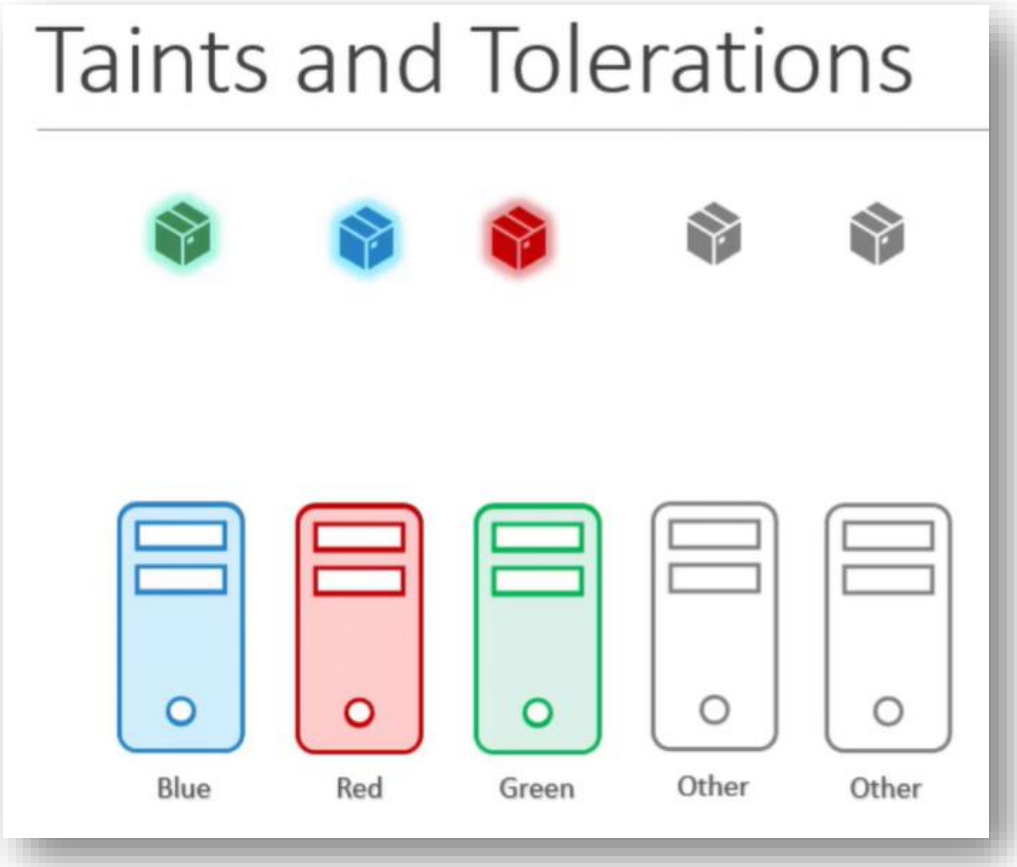
requiredDuringScheduling<u>Required</u>DuringExecution

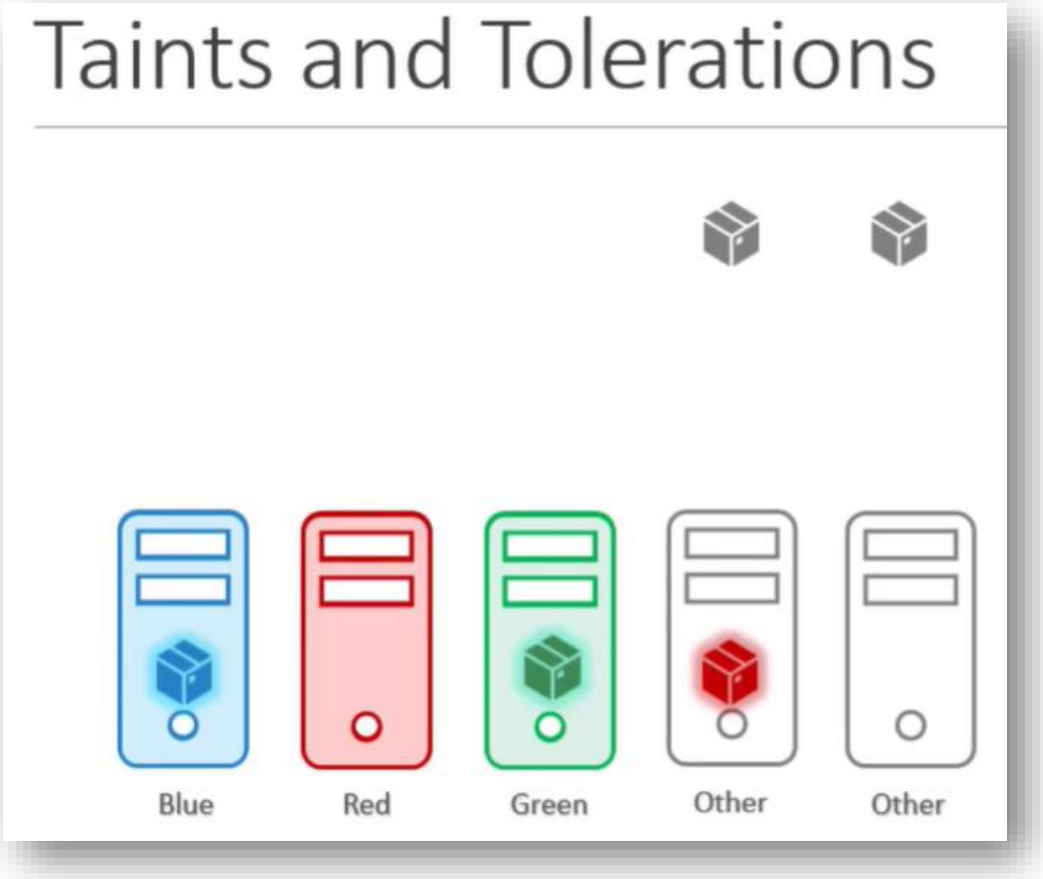
preferredDuringScheduling<u>Required</u>DuringExecution

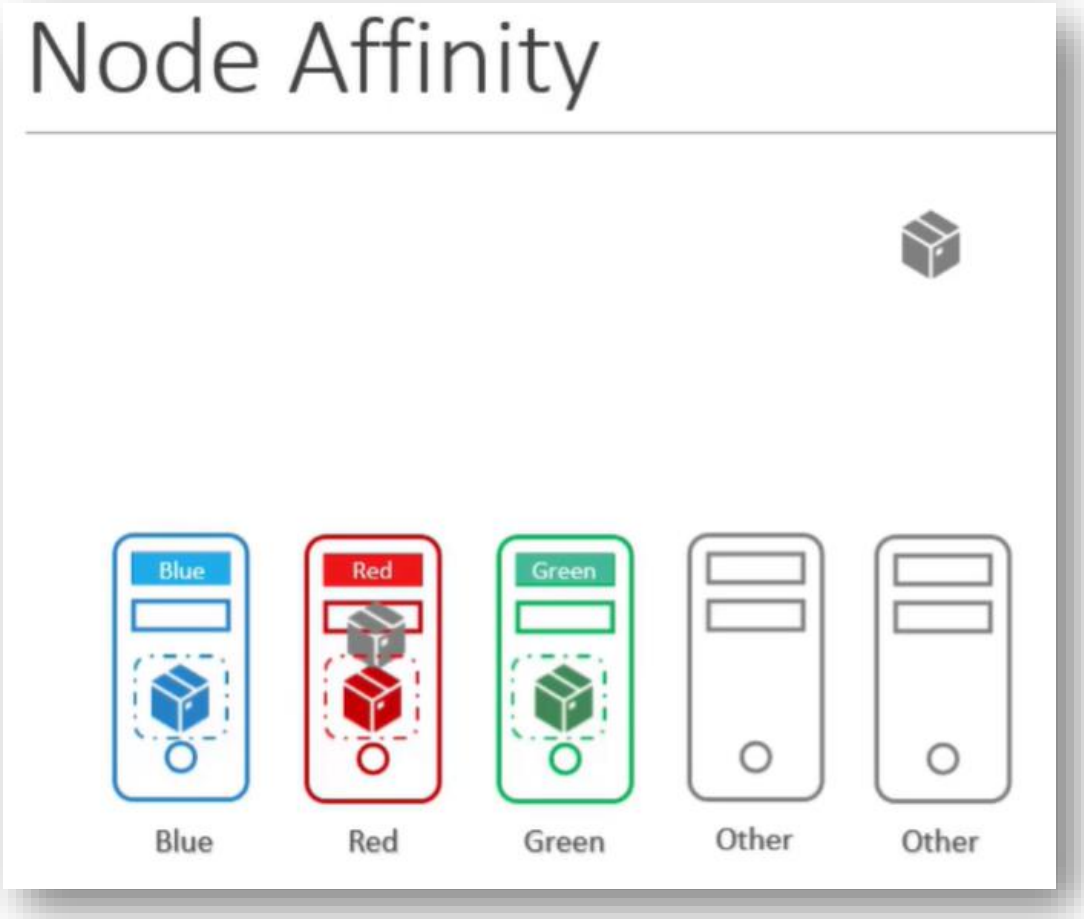
	DuringScheduling	DuringExecution
Type 1	Required	Ignored
Type 2	Preferred	Ignored
Type 3	Required	Required
Type 4	Preferred	Required

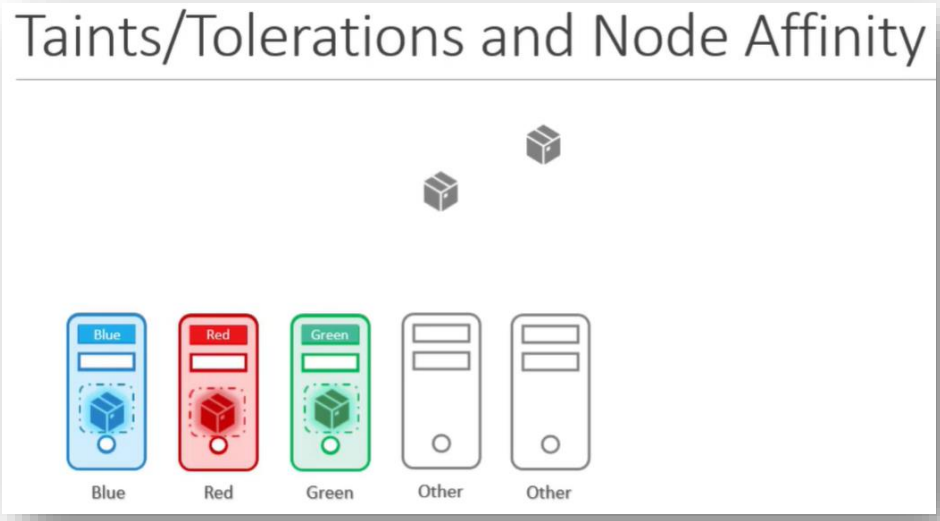
3.6 TAINTS AND TOLERATIONS V/S NODE AFFINITY






Taints and Tolerations
Blue
Red
Green
Other
Other


Node Affinity
Blue
Red
Green
Blue
Red
Green
Other
Other



3.7 RESOURCE LIMIT

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080
    resources:
      requests:
        memory: "4Gi"
        cpu: 2
```

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080
    resources:
      requests:
        memory: "1Gi"
        cpu: 1
      limits:
        memory: "2Gi"
        cpu: 2
```

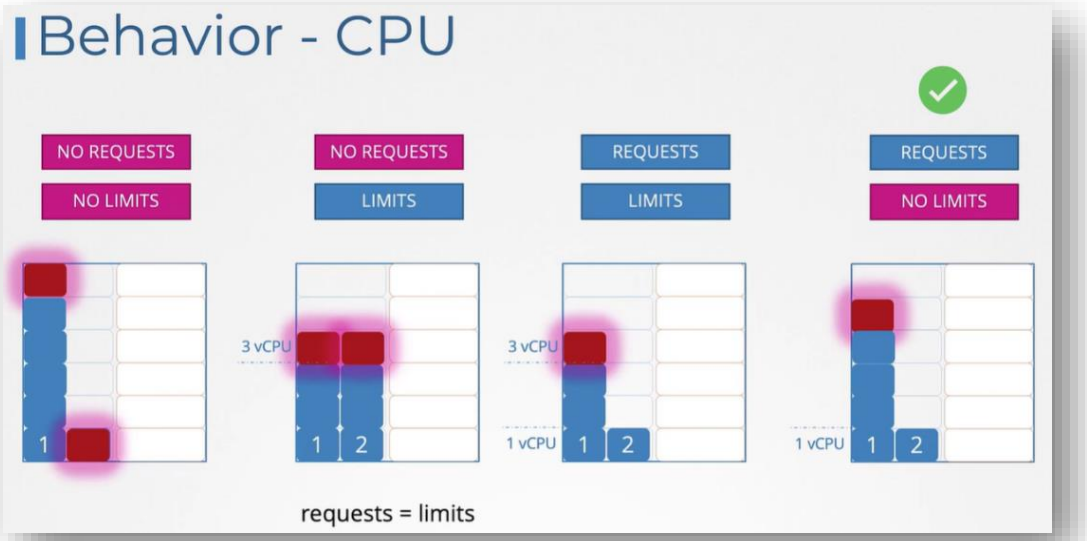
limit-range-cpu.yaml

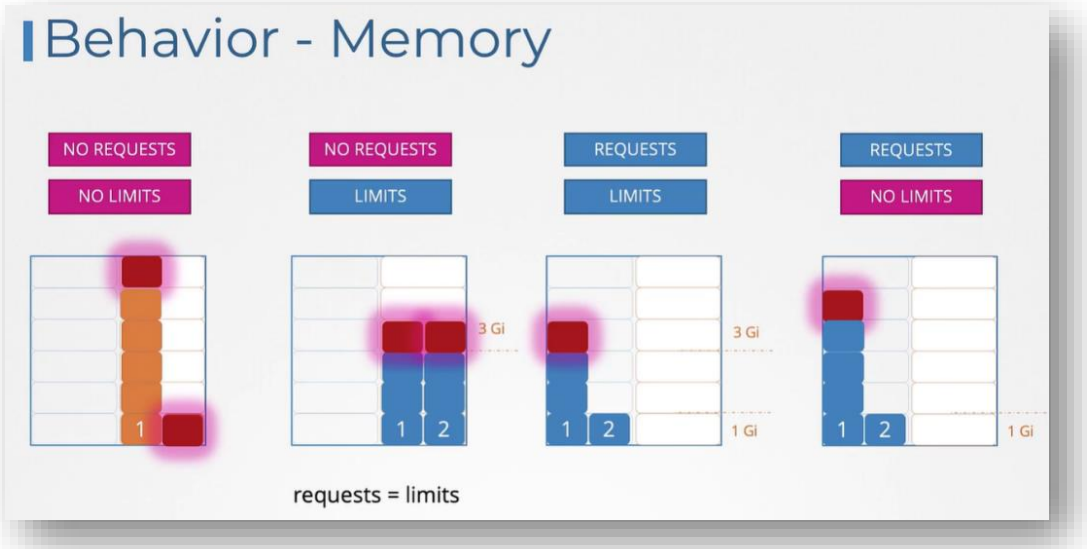
```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-resource-constraint
spec:
  limits:
  - default:
      cpu: 500m
    defaultRequest:
      cpu: 500m
    max:
      cpu: "1"
    min:
      cpu: 100m
    type: Container
```

limit

request

limit





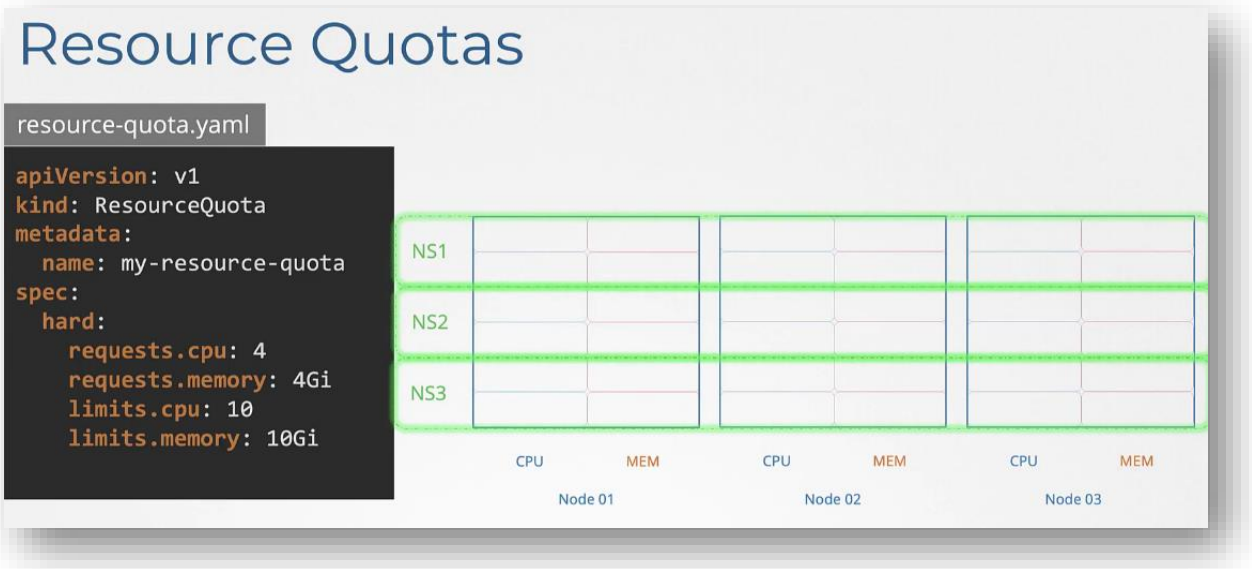
LimitRange

limit-range-cpu.yaml

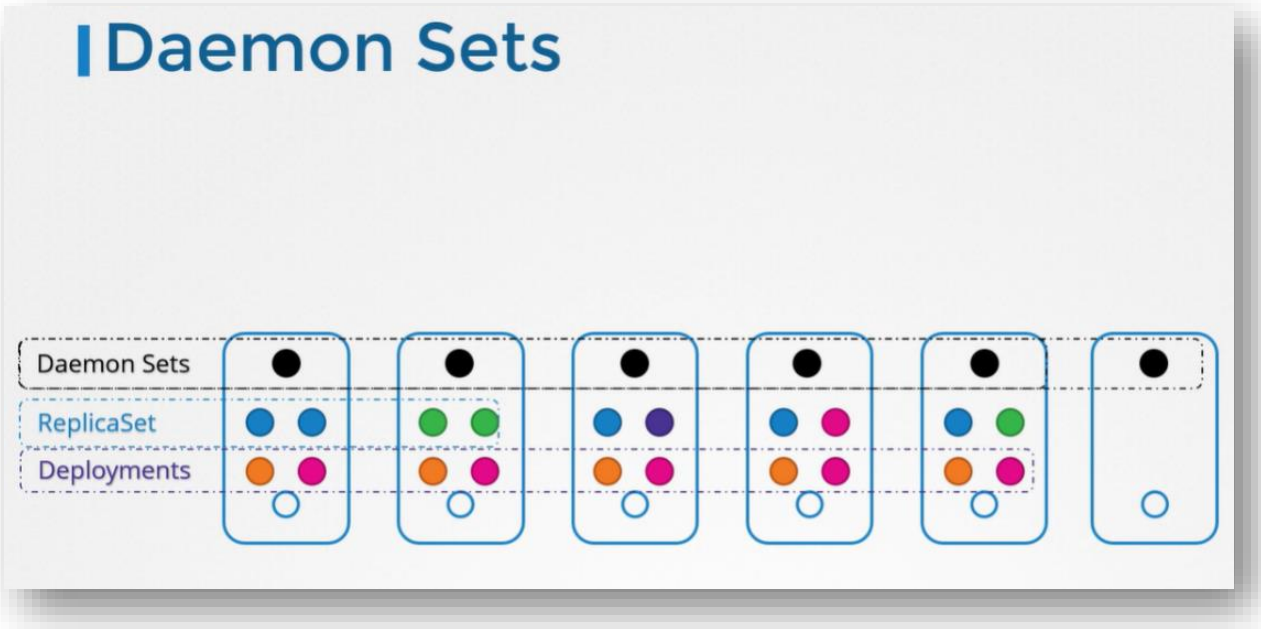
```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-resource-constraint
spec:
  limits:
  - default:
      cpu: 500m                    # limit
    defaultRequest:
      cpu: 500m                    # request
    max:
      cpu: "1"                     # limit
    min:
      cpu: 100m                    # request
    type: Container
```

limit-range-memory.yaml

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: memory-resource-constraint
spec:
  limits:
  - default:
      memory: 1Gi
    defaultRequest:
      memory: 1Gi
    max:
      memory: 1Gi
    min:
      memory: 500Mi
    type: Container
```



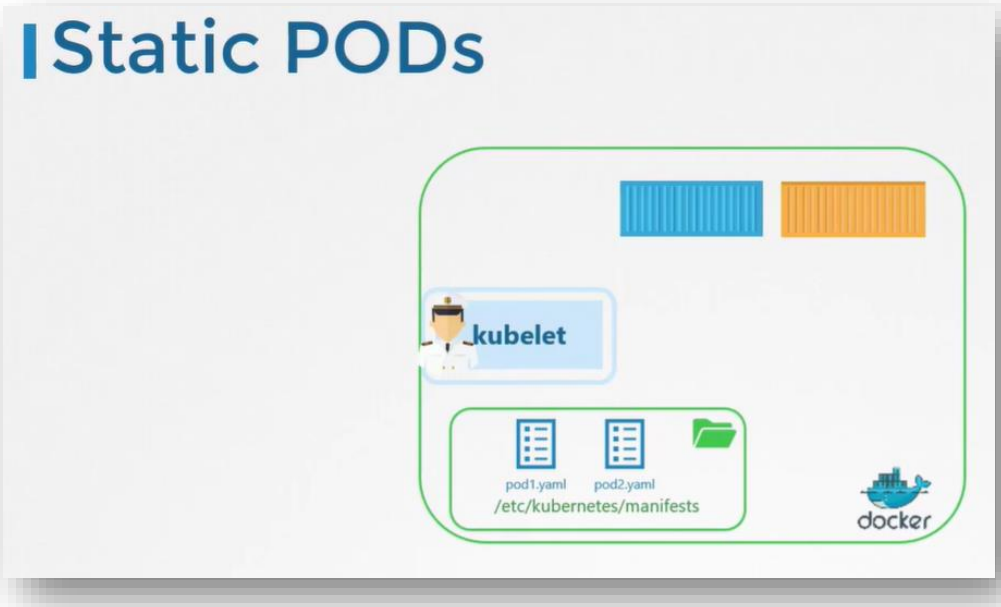
3.8 DAEMON SET



3.8.1 COMMAND LINE

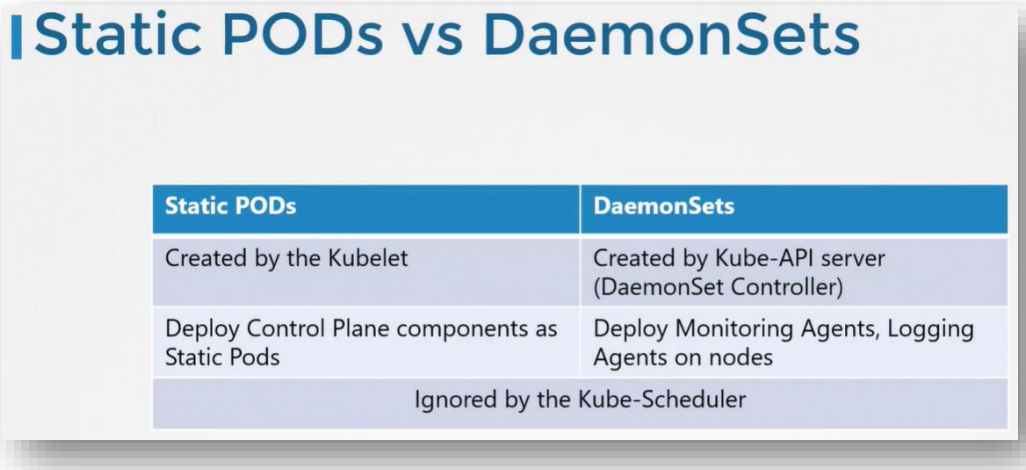
- kubectl get daemonset
- kubectl get daemonset -A
- kubectl describe daemonset daemonsetname
- kubectl describe daemonset kube-proxy --namespace=kube-system
- kubectl describe daemonset kube-flannel-ds --namespace=kube-flannel
- kubectl create deployment elasticsearch --image=registry.k8s.io/fluentd-elasticsearch:1.20 -n kube-system --dry-run=client -o yaml > fluentd.yaml

3.9 STATIC PODS




```
ExecStart=/usr/local/bin/kubelet \\
  --container-runtime=remote \\
  --container-runtime-endpoint=unix:///var/run/containerd/containerd.sock \\
  --pod-manifest-path=/etc/Kubernetes/manifests \\
  --kubeconfig=/var/lib/kubelet/kubeconfig \\
  --network-plugin=cni \\
  --register-node=true \\
  --v=2
```

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

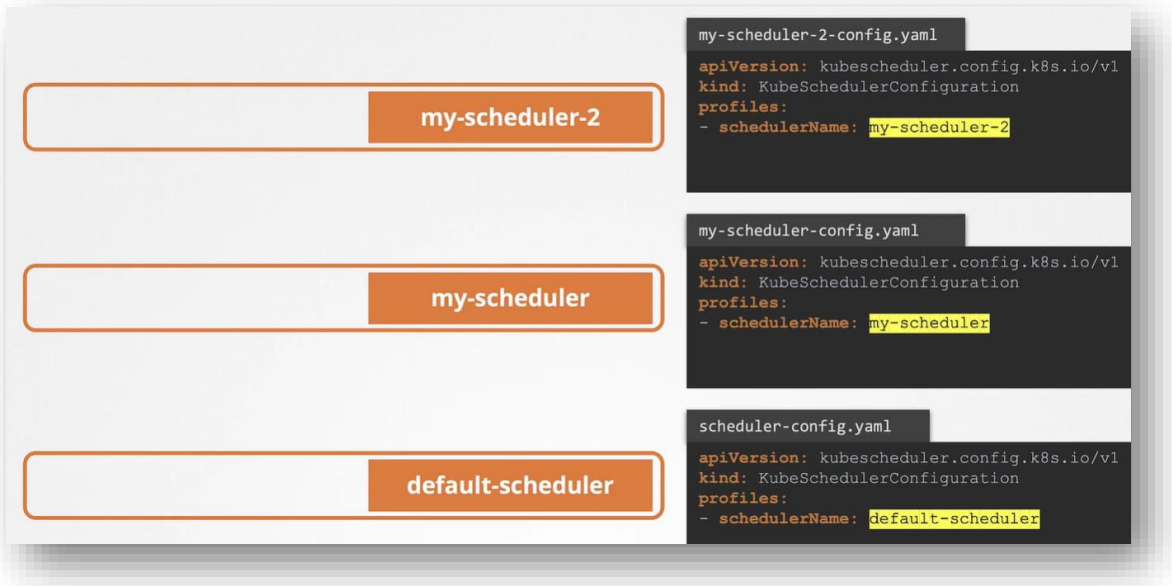


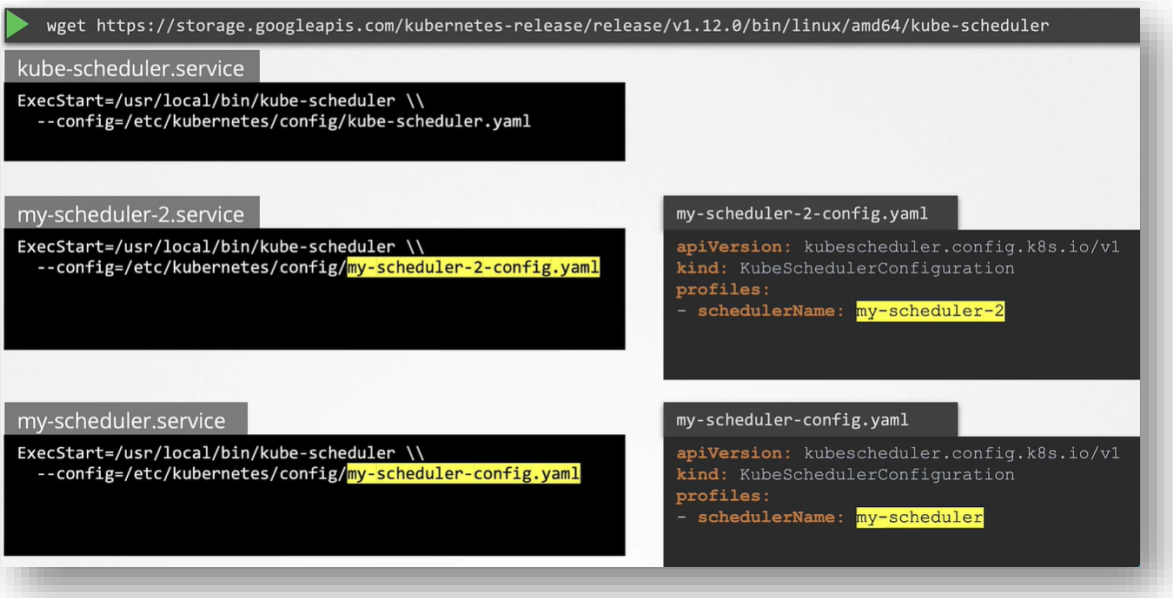
Static PODs	DaemonSets
Created by the Kubelet	Created by Kube-API server (DaemonSet Controller)
Deploy Control Plane components as Static Pods	Deploy Monitoring Agents, Logging Agents on nodes
Ignored by the Kube-Scheduler	

3.9.1 COMMANDLINE

- kubectl get pods -o wide
- kubectl get pods –all-namespaces
- kubectl get pods -A
- kubectl get pods -o wide --all-namespaces | grep controlplane
- kubectl run my-pod --image=nginx
- kubectl run static-busybox --image=busybox --restart=Never --command sleep 1000 --dry-run=client -o yaml > /etc/kubernetes/manifests/static-busybox.yaml
- /etc/Kubernetes/manifests (IMP)
- /var/lib/kubelet/config.yaml (IMP)

3.10 MULTIPLE SCHEDULERS





Deploy Additional Scheduler as a Pod

my-custom-scheduler.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-custom-scheduler
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-scheduler
    - --address=127.0.0.1
    - --kubeconfig=/etc/kubernetes/scheduler.conf
    - --config=/etc/kubernetes/my-scheduler-config.yaml
    image: k8s.gcr.io/kube-scheduler-amd64:v1.11.3
    name: kube-scheduler
```

my-scheduler-config.yaml

```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: my-scheduler
leaderElection:
  leaderElect: true
  resourceNamespace: kube-system
  resourceName: lock-object-my-scheduler
```

```
kubectl get pods --namespace=kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
coredns-78fcdf6894-bk4ml	1/1	Running	0	1h
coredns-78fcdf6894-ppr6m	1/1	Running	0	1h
etcd-master	1/1	Running	0	1h
kube-apiserver-master	1/1	Running	0	1h
kube-controller-manager-master	1/1	Running	0	1h
kube-proxy-dgbgv	1/1	Running	0	1h
kube-proxy-fptbr	1/1	Running	0	1h
kube-scheduler-master	1/1	Running	0	1h
my-custom-scheduler	1/1	Running	0	9s
weave-net-4tfpt	2/2	Running	1	1h
weave-net-6j6zs	2/2	Running	1	1h

Use Custom Scheduler

```
kubectl get pods --namespace=kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
coredns-78fcdf6894-bk4ml	1/1	Running	0	1h
coredns-78fcdf6894-ppr6m	1/1	Running	0	1h
etcd-master	1/1	Running	0	1h
kube-apiserver-master	1/1	Running	0	1h
kube-controller-manager-master	1/1	Running	0	1h
kube-proxy-dgbgv	1/1	Running	0	1h
kube-proxy-fptbr	1/1	Running	0	1h
kube-scheduler-master	1/1	Running	0	1h
my-custom-scheduler	1/1	Running	0	9s
weave-net-4tfpt	2/2	Running	1	1h
weave-net-6j6zs	2/2	Running	1	1h

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
  schedulerName: my-custom-scheduler
```

3.10.1 COMMANDLINE

- kubectl get events -o wide
- kubectl logs my-custom-scheduler --name-space=kube-system
- kubectl get pods -n kube-system
- kubectl get pods -n kube-system -o wide -A
- kubectl describe pod kube-scheduler-controlplane -n kube-system
- kubectl get sa my-scheduler -n kube-system (ServiceAccount)
- kubectl create configmap my-scheduler-config –from-file=path -n kube-system
- kubectl get configmap my-scheduler-config -n kube-system

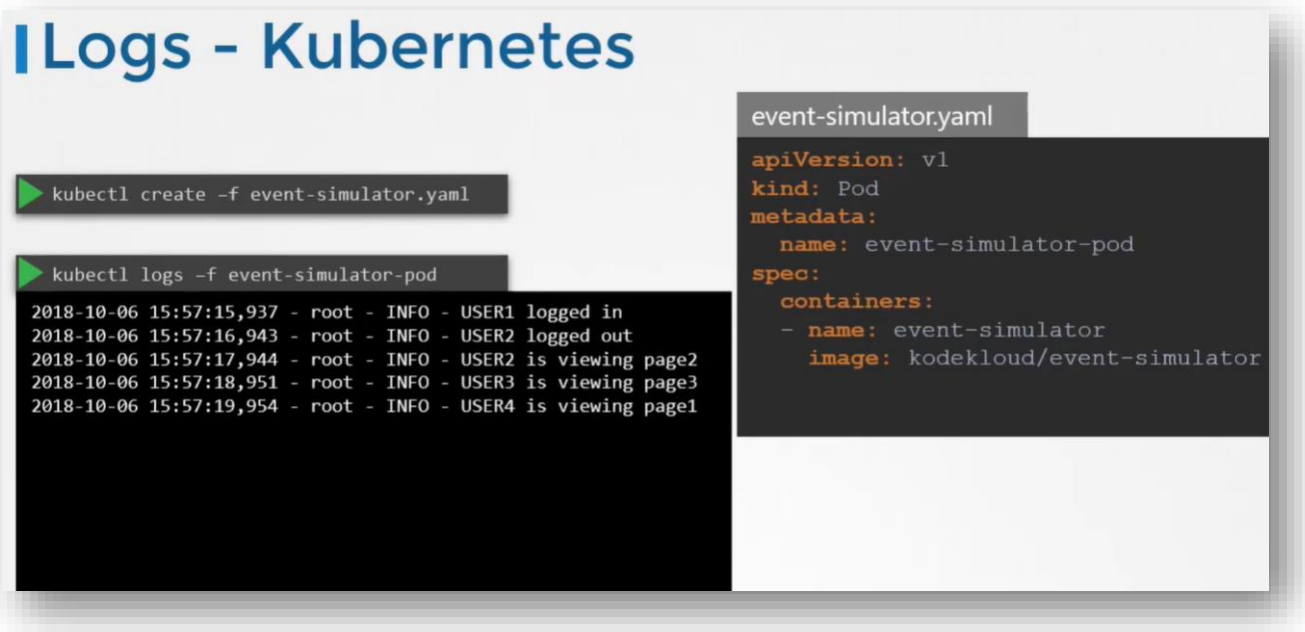
4. LOGGING AND MONITORING

4.1 MONITOR CLUSTER COMPONENTS

4.1.1 COMMANDLINE

- kubectl top node
- kubectl top pod
- kubectl create -f .
- kubectl top nodes --sort-by='cpu' --no-headers | head -1
- kubectl top pods --sort-by='cpu' --no-headers | head -1

4.2 MANAGING APPLICATIONS LOGS



event-simulator.yaml

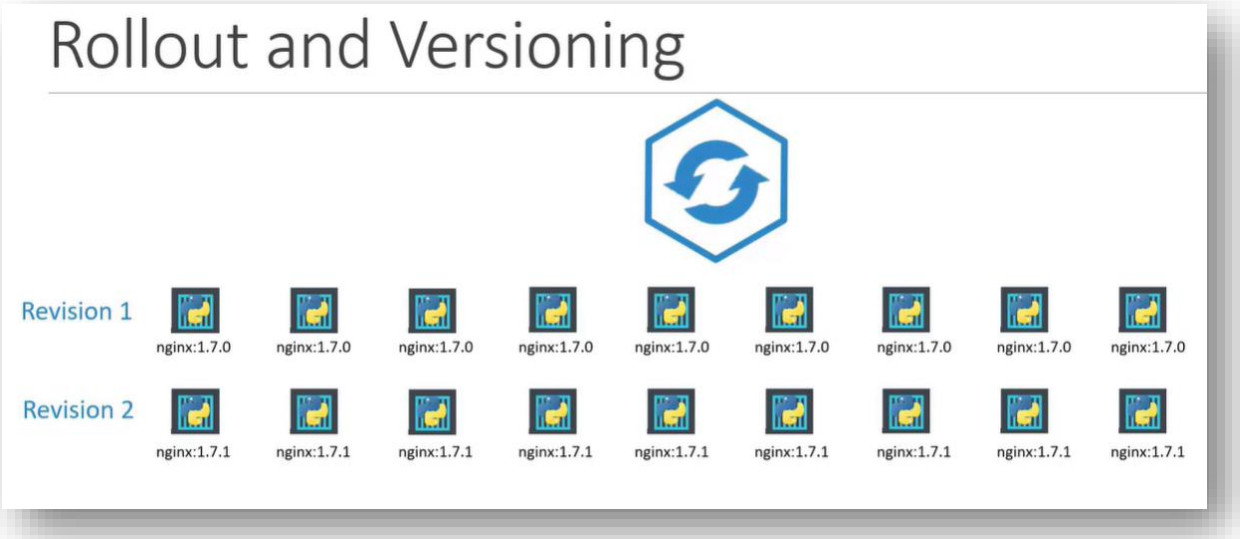
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: event-simulator-pod
spec:
  containers:
  - name: event-simulator
    image: kodekloud/event-simulator
  - name: image-processor
    image: some-image-processor
```

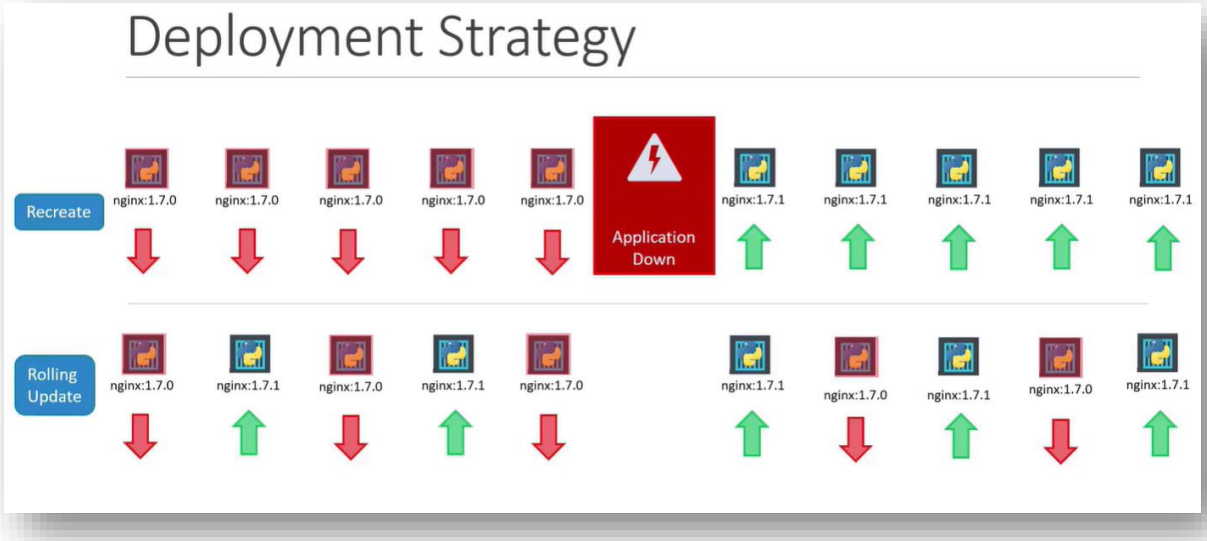
4.2.1 COMMANDLINE

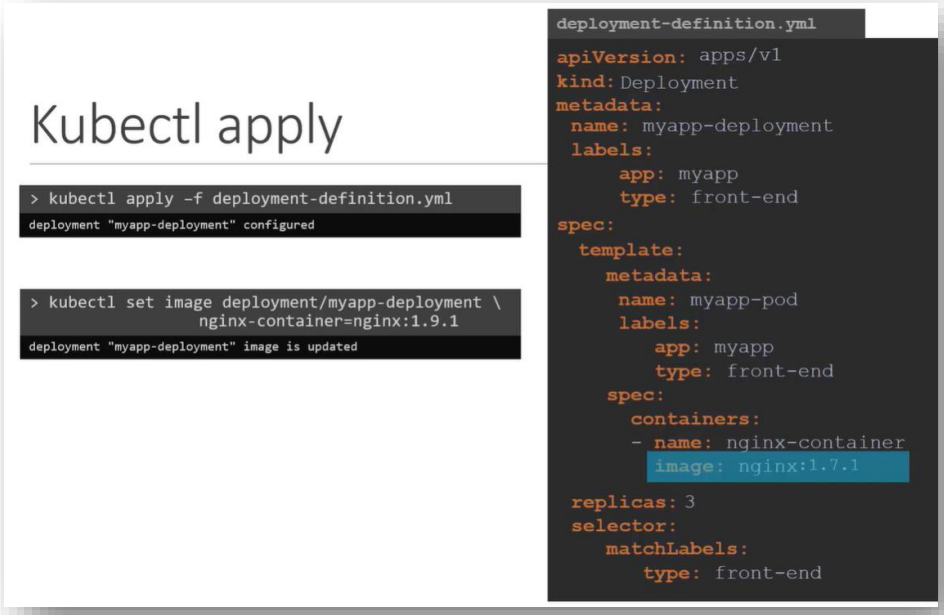
- kubectl logs webapp-1 | grep USER5
- kubectl logs webapp-2 -c simple-webapp

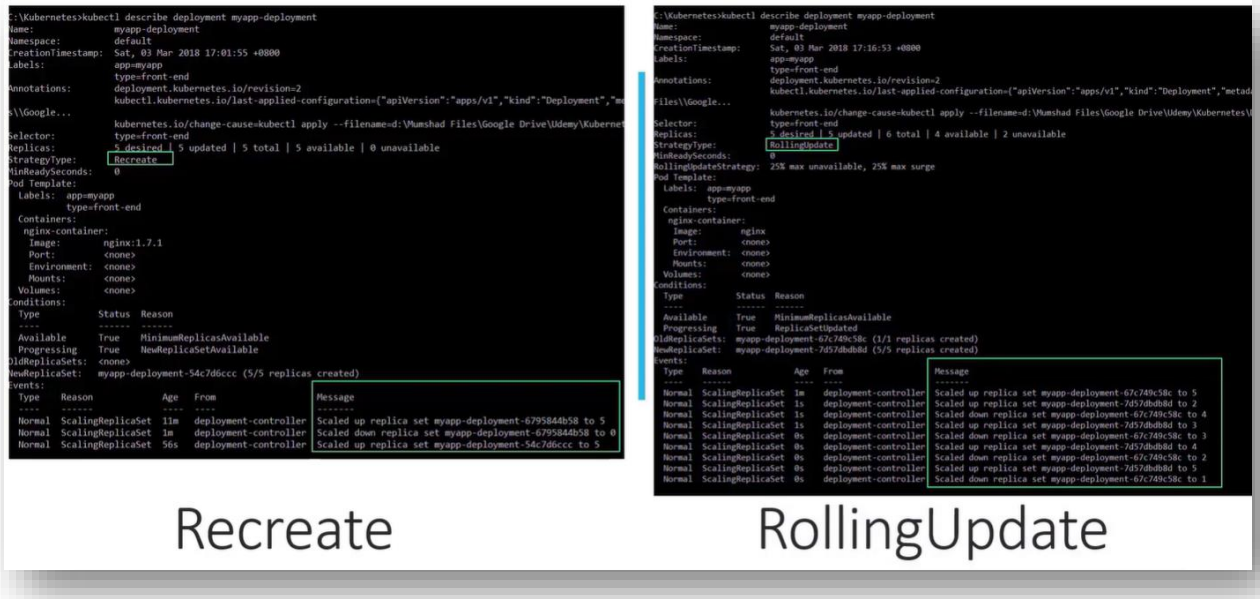
5. APPLICATIONS LIFECYCLE MANAGEMENT

5.1 ROLLING UPDATES AND ROLLBACKS

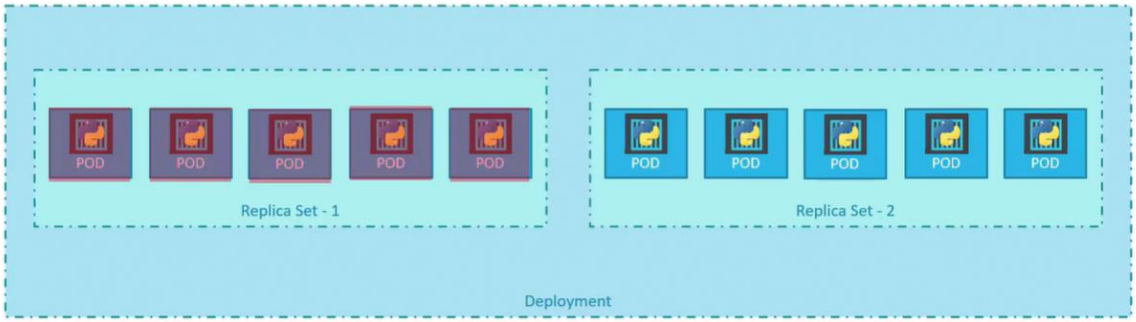



Deployment Strategy
Recreate
nginx:1.7.0
nginx:1.7.0
nginx:1.7.0
nginx:1.7.0
nginx:1.7.0
Application Down
nginx:1.7.1
nginx:1.7.1
nginx:1.7.1
nginx:1.7.1
nginx:1.7.1
Rolling Update
nginx:1.7.0
nginx:1.7.1
nginx:1.7.0
nginx:1.7.1
nginx:1.7.0
nginx:1.7.1
nginx:1.7.0
nginx:1.7.1
nginx:1.7.0
nginx:1.7.1


Kubectl apply
> kubectl apply -f deployment-definition.yml
deployment "myapp-deployment" configured
> kubectl set image deployment/myapp-deployment \
nginx-container=nginx:1.9.1
deployment "myapp-deployment" image is updated
deployment-definition.yml
apiVersion: apps/v1
kind: Deployment
metadata:
name: myapp-deployment
labels:
app: myapp
type: front-end
spec:
template:
metadata:
name: myapp-pod
labels:
app: myapp
type: front-end
spec:
containers:
- name: nginx-container
image: nginx:1.7.1
replicas: 3
selector:
matchLabels:
type: front-end


Recreate
RollingUpdate

Upgrades

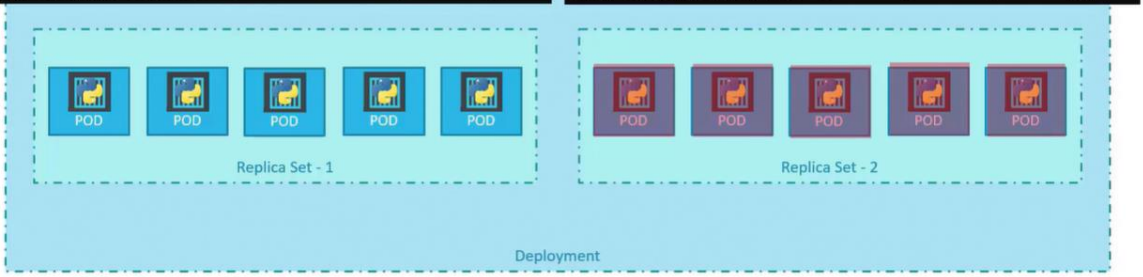



```
> kubectl get replicasets
NAME                          DESIRED   CURRENT   READY   AGE
myapp-deployment-67c749c58c   0         0         0       22m
myapp-deployment-7d57dbdb8d   5         5         5       20m
```

Rollback

```
> kubectl get replicasets
NAME                          DESIRED   CURRENT   READY   AGE
myapp-deployment-67c749c58c   0         0         0       22m
myapp-deployment-7d57dbdb8d   5         5         5       20m
```

```
> kubectl get replicasets
NAME                          DESIRED   CURRENT   READY   AGE
myapp-deployment-67c749c58c   5         5         5       22m
myapp-deployment-7d57dbdb8d   0         0         0       20m
```

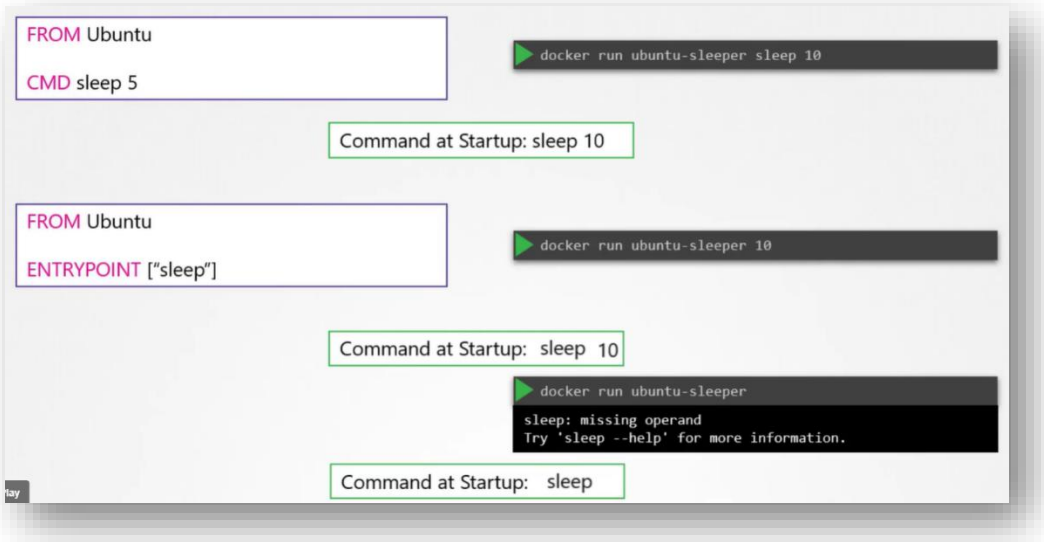


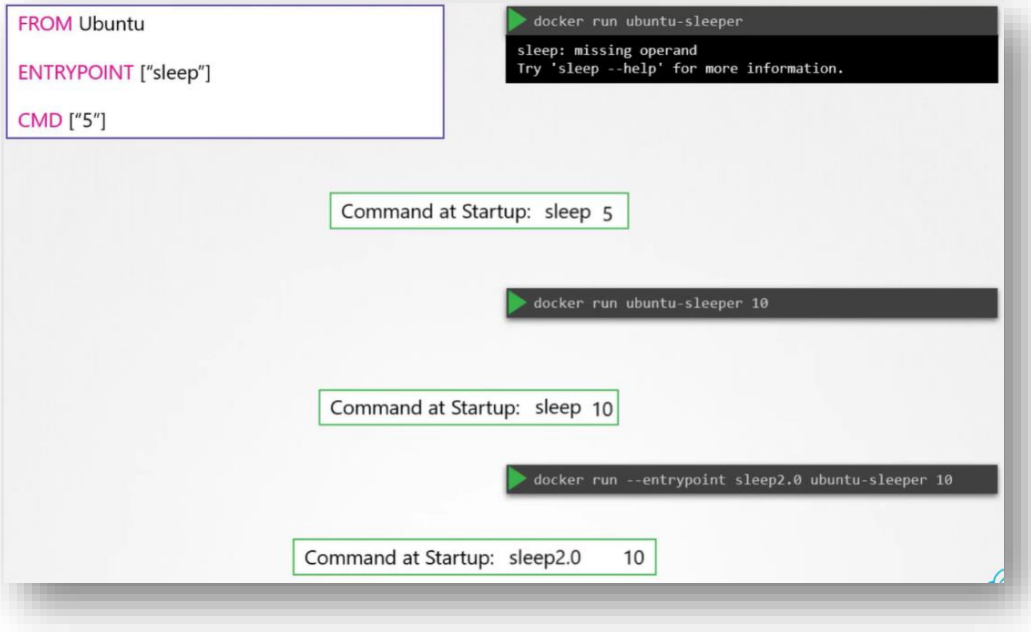

```
> kubectl rollout undo deployment/myapp-deployment
deployment "myapp-deployment" rolled back
```

Summarize Commands

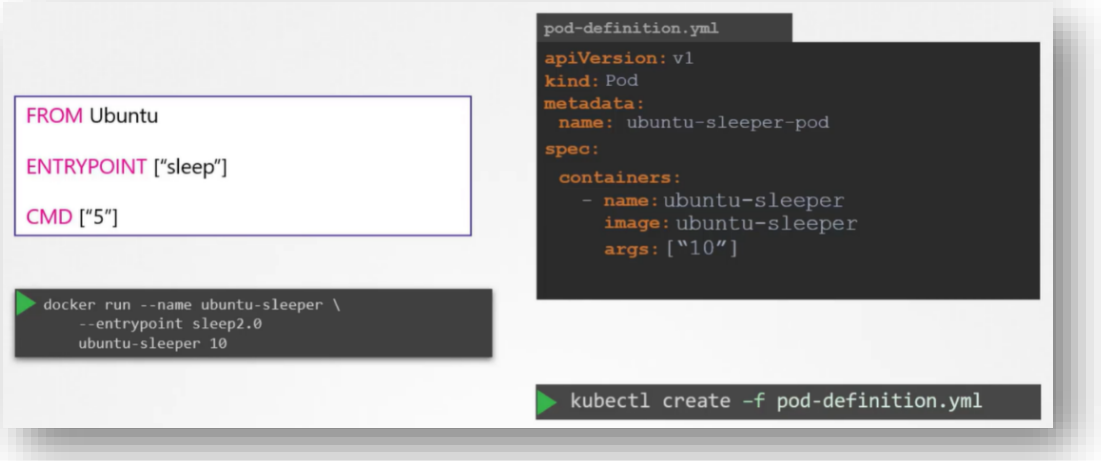
Create	`> kubectl create -f deployment-definition.yml`
Get	`> kubectl get deployments`
Update	`> kubectl apply -f deployment-definition.yml`
	`> kubectl set image deployment/myapp-deployment nginx=nginx:1.9.1`
Status	`> kubectl rollout status deployment/myapp-deployment`
	`> kubectl rollout history deployment/myapp-deployment`
Rollback	`> kubectl rollout undo deployment/myapp-deployment`

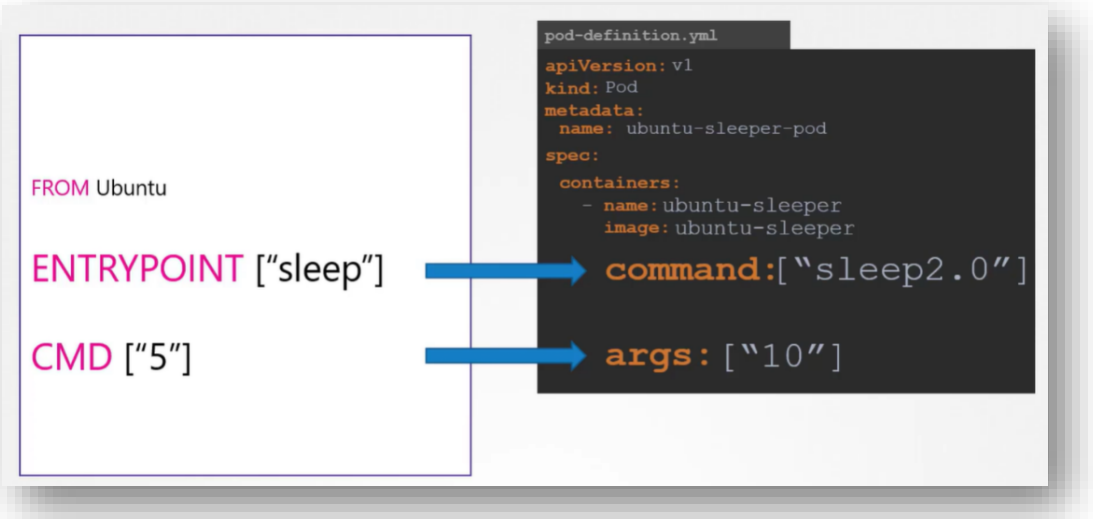
5.2 COMMANDS AND ARGUMENTS IN DOCKER





5.3 COMMANDS AND ARGUMENTS IN KUBERNETES






```yaml
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-sleeper-2
spec:
  containers:
  - name: ubuntu
    image: ubuntu
    command: [ "sleep" ]
    args: [ "5000" ]
```

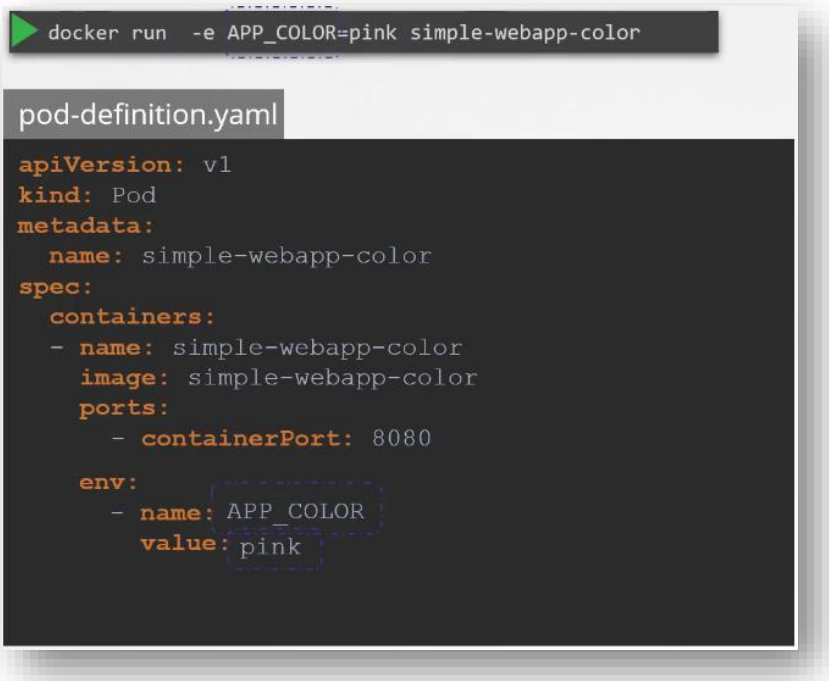
```yaml
kind: Pod
metadata:
  name: webapp-green
  labels:
      name: webapp-green
spec:
  containers:
  - name: simple-webapp
    image: kodekloud/webapp-color
    command: ["--color","green"]
```

```
FROM python:3.6-alpine

RUN pip install flask

COPY . /opt/

EXPOSE 8080

WORKDIR /opt

ENTRYPOINT ["python", "app.py"]

CMD ["--color", "red"]

controlplane ~ ➜  cat webapp-color-3/webapp-color-pod-2.yaml
apiVersion: v1
kind: Pod
metadata:
  name: webapp-green
  labels:
      name: webapp-green
spec:
  containers:
  - name: simple-webapp
    image: kodekloud/webapp-color
    command: ["python", "app.py"]
    args: ["--color", "pink"]
```

5.3.1 COMMANDLINE

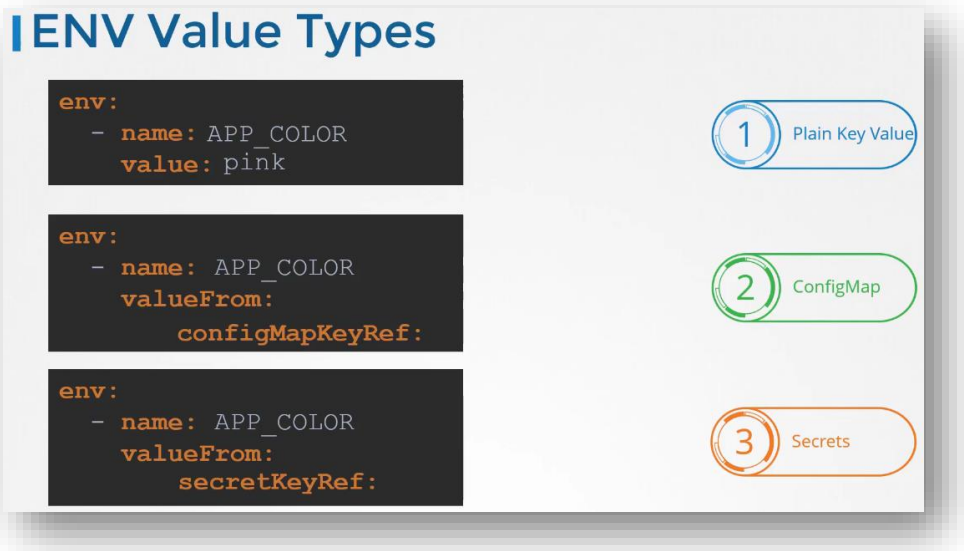
- kubectl describe pod ubuntu-sleeper
- kubectl replace --force -f ubuntu-sleeper.yaml
- kubectl run pod ubuntu --image=ubuntu --command sleep 5000
- kubectl run webapp-green --image=docker/webapp-color --command color green
- kubectl run weapp-green --image=docker/webapp-color -- --color green

5.4 CONFIGURE ENVIRONMENT VARIABLES IN APPLICATIONS

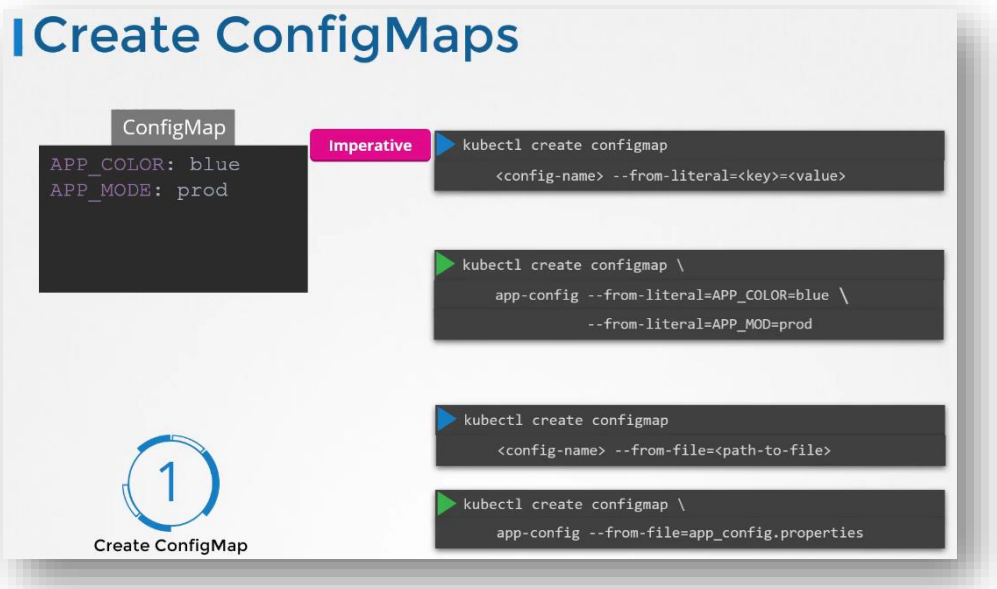


config-map.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  APP_COLOR: blue
  APP_MODE: prod
```



5.5 CONFIGURE CONFIG MAP IN APPLICATIONS



Create ConfigMaps

app-config

```
APP_COLOR: blue
APP_MODE: prod
```

mysql-config

```
port: 3306
max_allowed_packet: 128M
```

redis-config

```
port: 6379
rdb-compression: yes
```

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080
    envFrom:
      - configMapRef:
          name: app-config
```

ConfigMap in Pods

```yaml
envFrom:
  - configMapRef:
      name: app-config
```

ENV

SINGLE ENV

```yaml
env:
  - name: APP_COLOR
    valueFrom:
      configMapKeyRef:
        name: app-config
        key: APP_COLOR
```

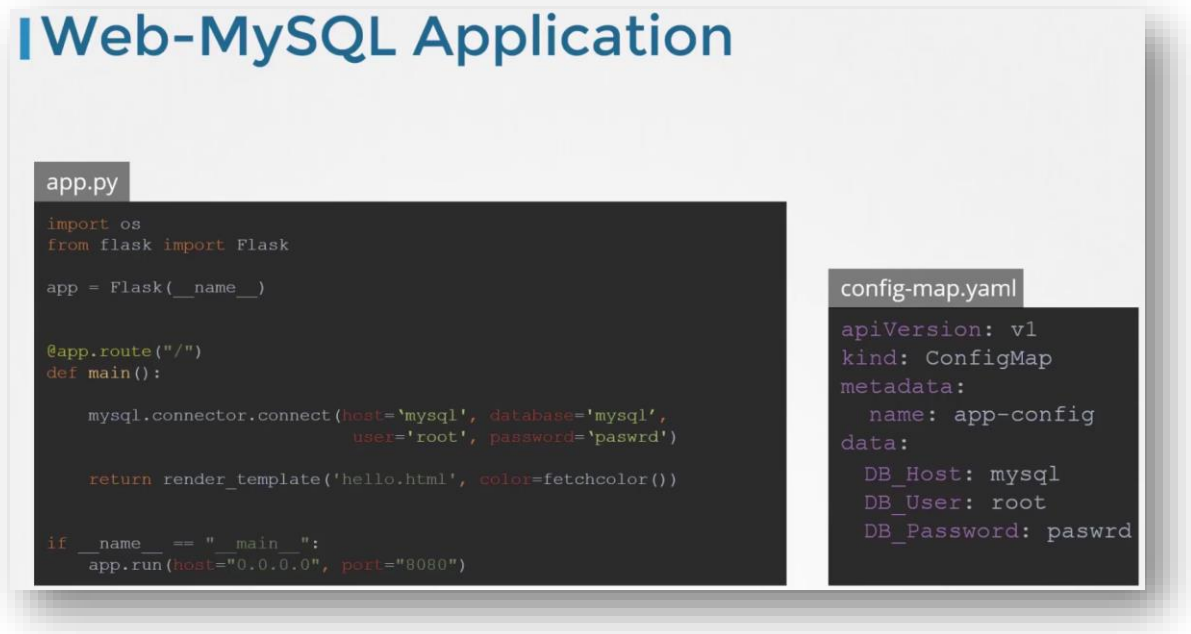
```yaml
volumes:
- name: app-config-volume
  configMap:
    name: app-config
```

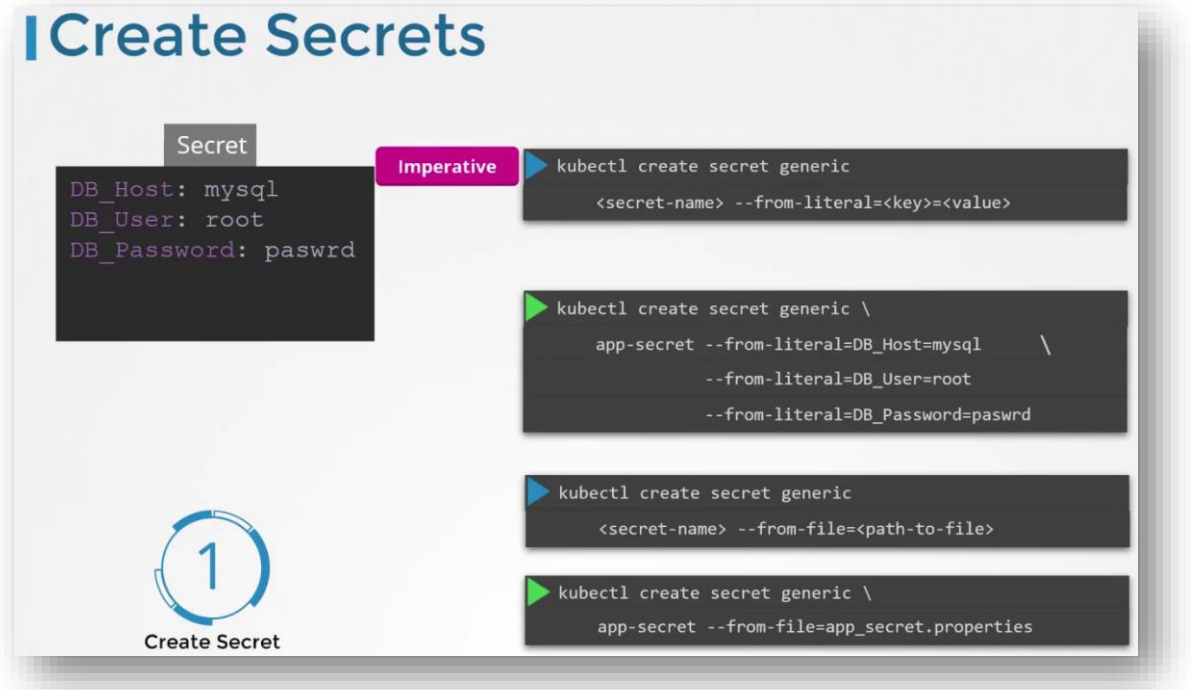
VOLUME

5.5.1 COMMANDLINE

- kubectl create configmap webapp-config-map --from-literal=APP_COLOR=darkblue --from-literal=APP_OTHER=disregard
- kubectl create configmap webapp-config-map --from-literal=APP_COLOR=darkblue
- kubectl get configmaps
- kubectl describe configmaps
- kubectl replace --force -f /tmp/kubectl-edit-4257439986.yaml

5.6 SECRETS





Encode Secrets

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

```
DB_Host: bXlzcWw=
DB_User: cm9vdA==
DB_Password: cGFzd3Jk
```

```
echo -n 'mysql' | base64
bXlzcWw=
```

```
echo -n 'root' | base64
cm9vdA==
```

```
echo -n 'paswrd' | base64
cGFzd3Jk
```

Decode Secrets

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

```
DB_Host: bXlzcWw=
DB_User: cm9vdA==
DB_Password: cGFzd3Jk
```

```
echo -n 'bXlzcWw=' | base64 --decode
mysql
```

```
echo -n 'cm9vdA==' | base64 --decode
root
```

```
echo -n 'cGFzd3Jk' | base64 --decode
paswrd
```

Secrets in Pods

```
envFrom:
  - secretRef:
      name: app-config
```

ENV

SINGLE ENV

```
env:
  - name: DB_Password
    valueFrom:
      secretKeyRef:
        name: app-secret
        key: DB_Password
```

```
volumes:
- name: app-secret-volume
  secret:
    secretName: app-secret
```

VOLUME

Secrets in Pods as Volumes

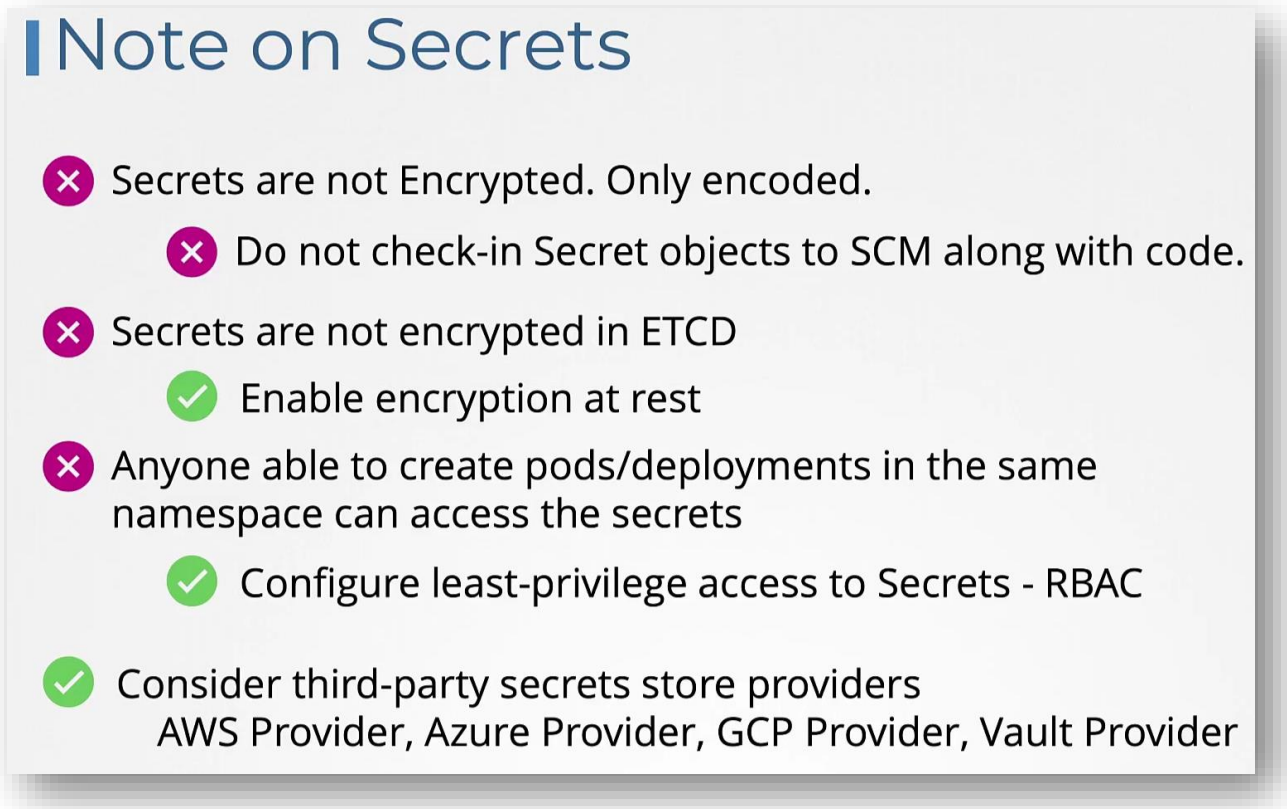
```
volumes:
- name: app-secret-volume
  secret:
    secretName: app-secret
```

VOLUME

```
ls /opt/app-secret-volumes
DB_Host      DB_Password  DB_User
```

```
cat /opt/app-secret-volumes/DB_Password
paswrd
```

Inside the Container



5.6.1 COMMANDLINE

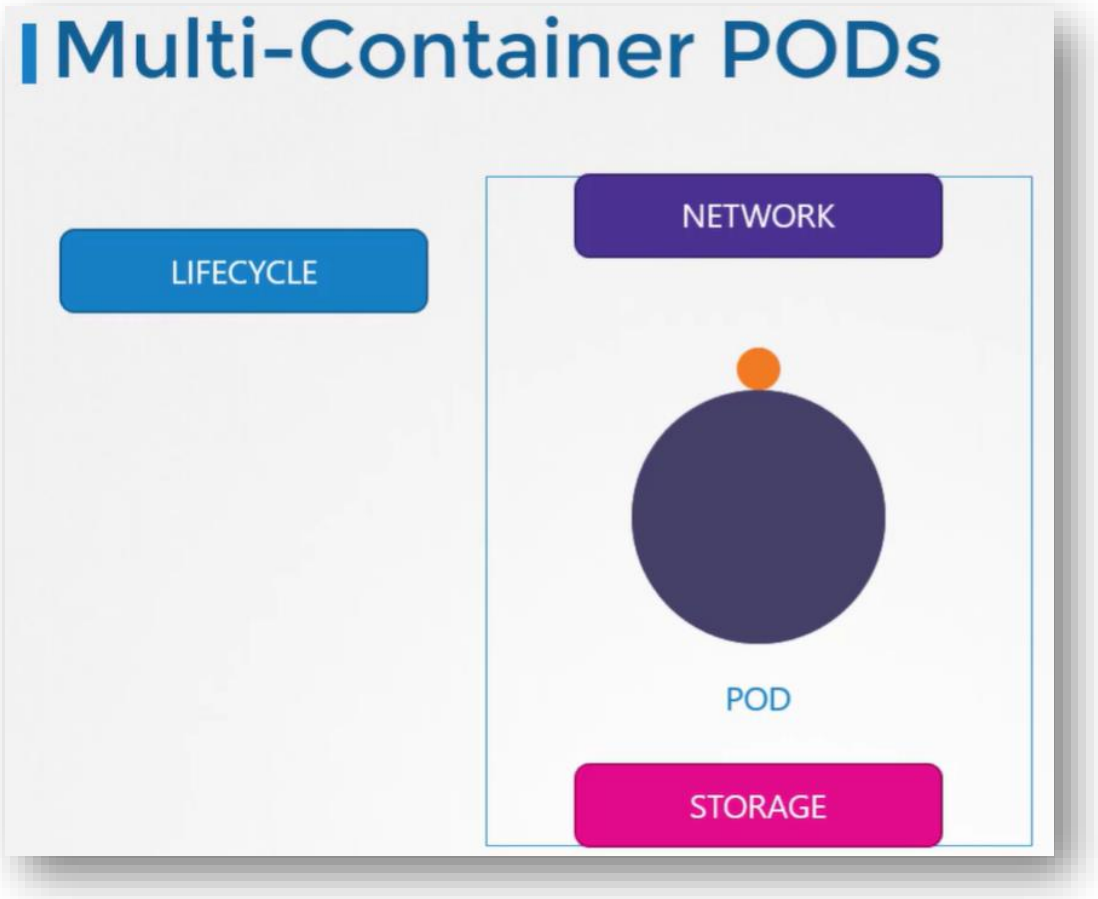
- kubectl get secrets
- kubectl describe secrets
- kubectl get secret app-secret -o yaml
- kubectl create secret generic secret_name --from-literal=key=value
- kubectl create secret generic my-secret --from-literal=DB_Host=sql01 --from-literal=DB_User=root --from-literal=DB_Password=password123
-
- kubectl get secret my-secret -o YAML (Check Argument)
- echo “Key1VALUe” | base64 --decode (Check Value Decoded)
- kubectl get pod -n kube-system

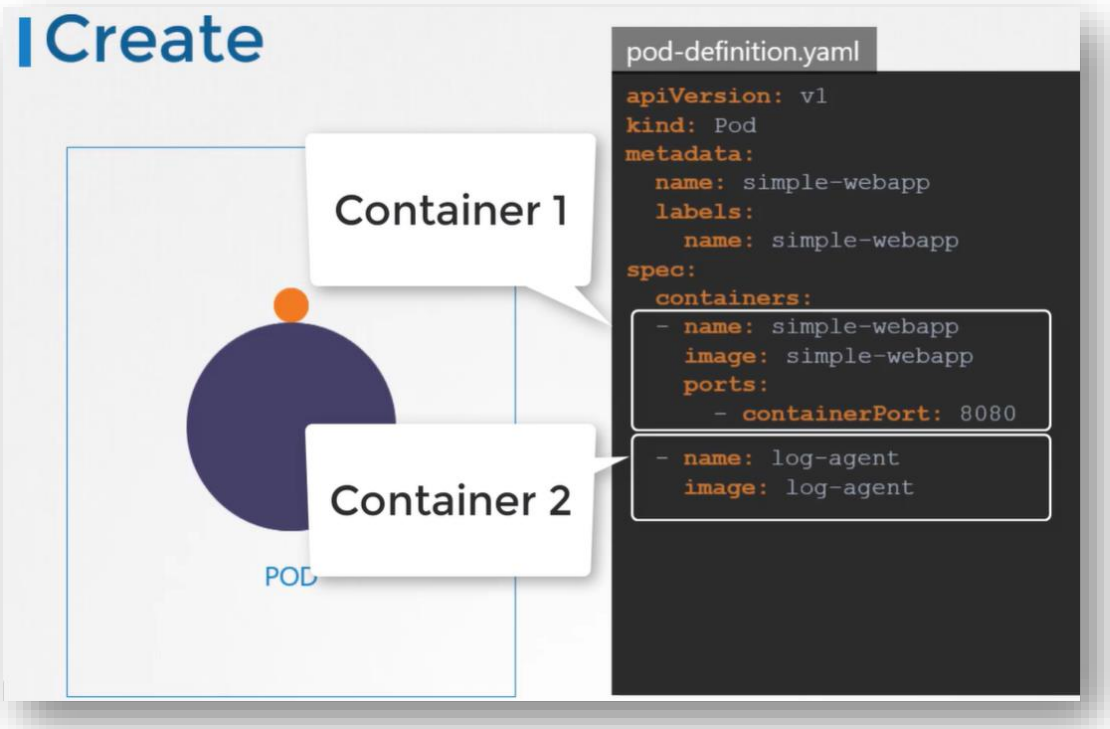
5.7 ENCRYPTING SECRET DATA AT REST

5.7.1 COMMANDLINE

- ps aux | grep kube-api
- ps aux | grep kube-api | grep encry
- crictl pods

5.8 MULTI CONTAINER PODS





5.8.1 COMMANDLINE

- kubectl -n elastic-stack exec -it app -- cat /log/app.log
- kubectl log -n elastic-stack app

5.9 MULTI CONTAINER PODS DESIGN PATTERNS

5.10 INIT CONTAINERS

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox:1.28
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
  initContainers:
  - name: init-myservice
    image: busybox
    command: ['sh', '-c', 'git clone  ;']
```

5.10.1 COMMANDLINE

- kubectl log orange -c init-myservice

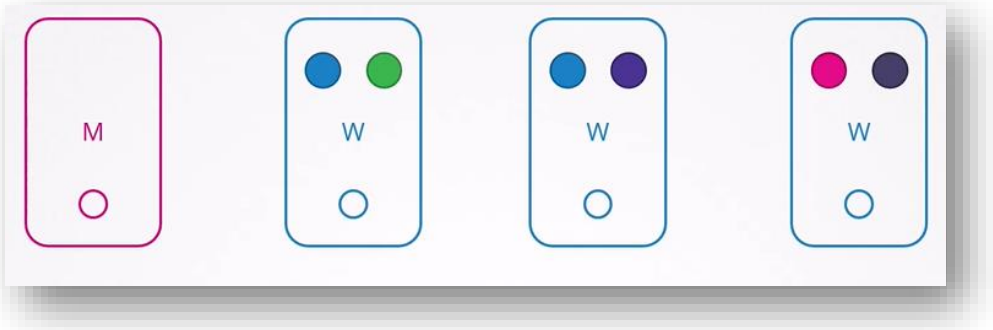
5.11 SELF HEALING APPLICATIONS

6. CLUSTER MAINTENANCE

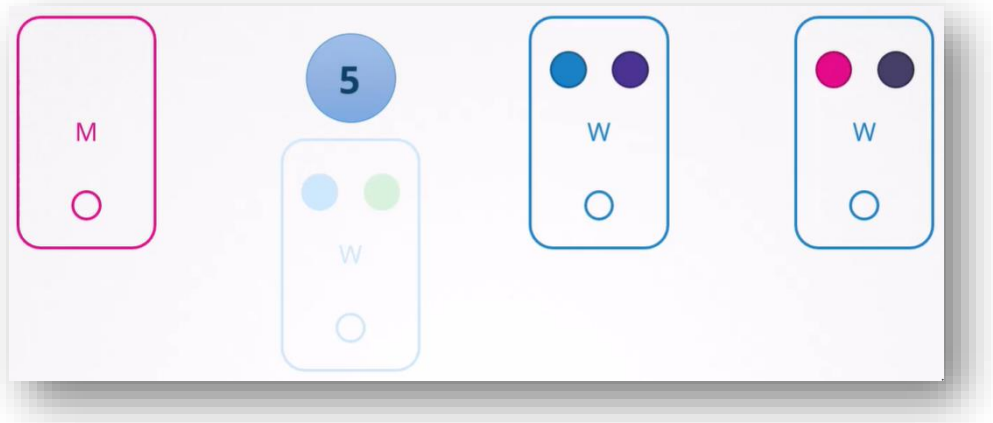
6.1 CLUSTER MAINTENANCE – SECTIONS INTRODUCTION

6.2 OPERATING SYSTEM UPGRADES

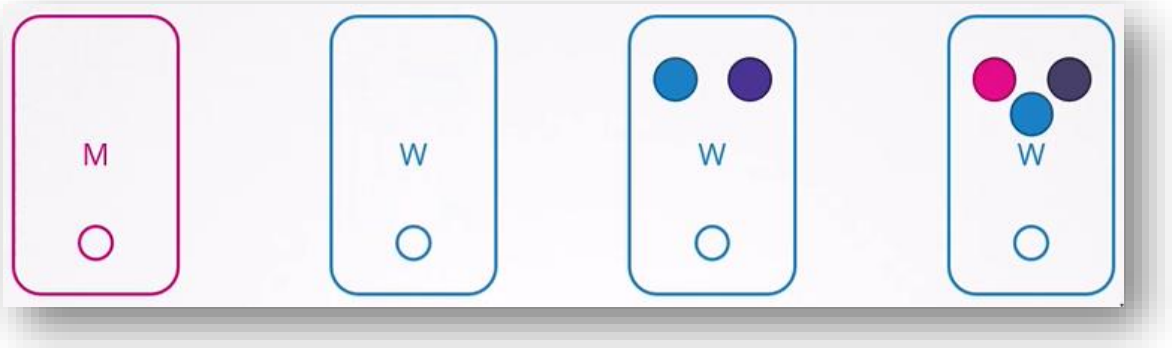
- Pod UP



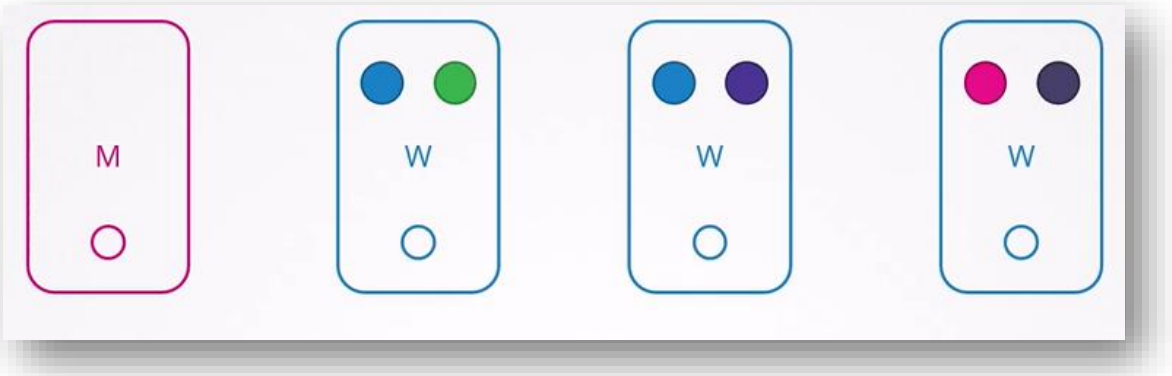
- Pod Down.



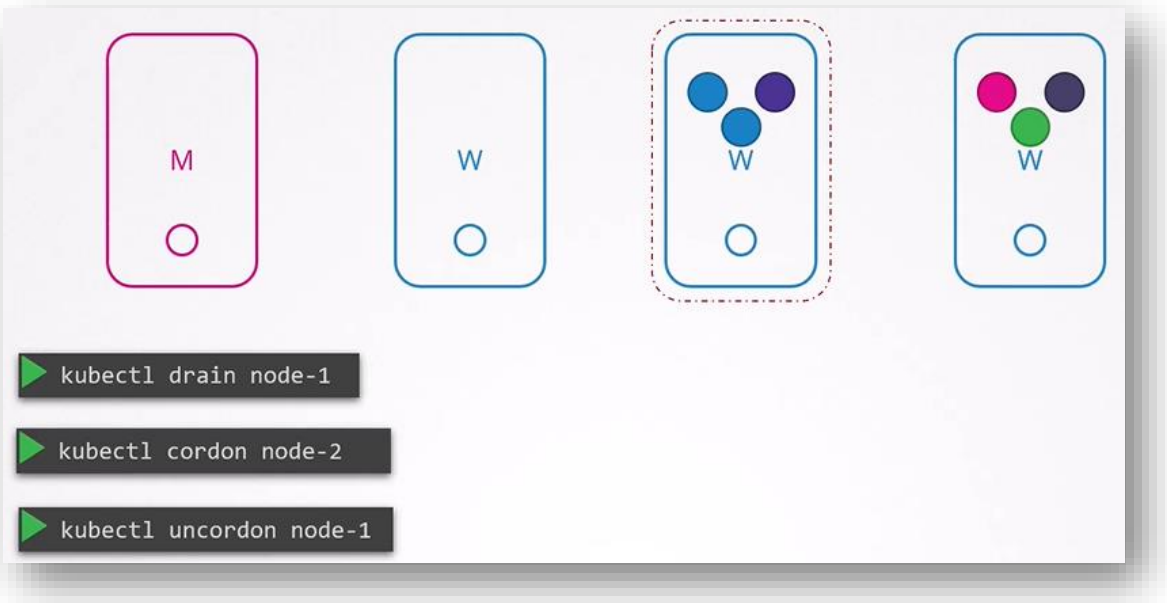
- Pod Recreate.



- Pod UP again



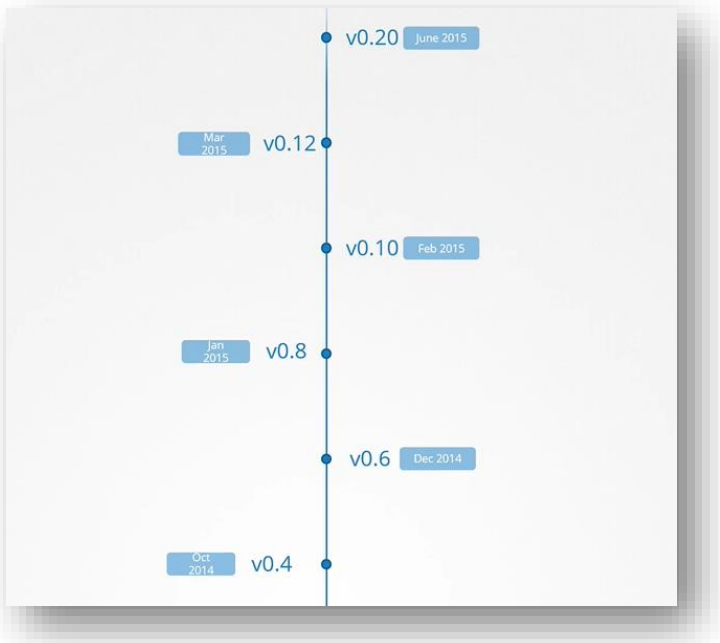
- Pod drain

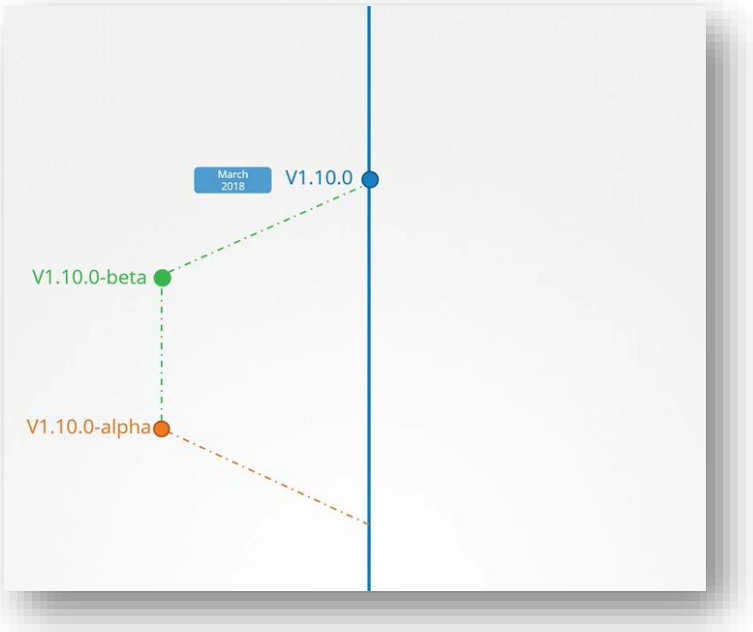


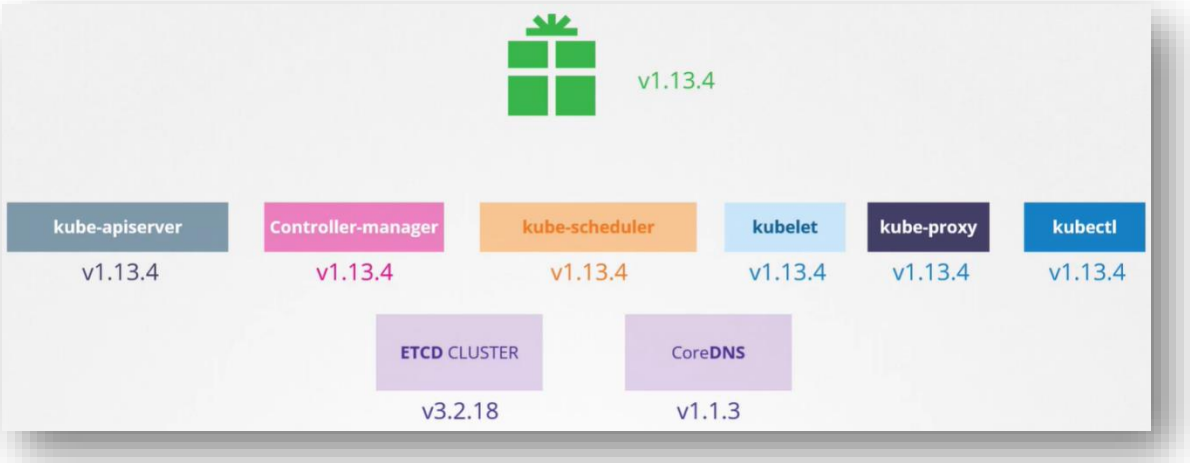
6.2.1 COMMANDLINE

- kubectl drain node01 (Move pod on other nodes).
- Kubectl cordon node01 (Mark a node unscheduled)
- Kubectl uncordon node01 (Mark a node Schedule)
- kubectl drain node01 --ignore-daemonsets

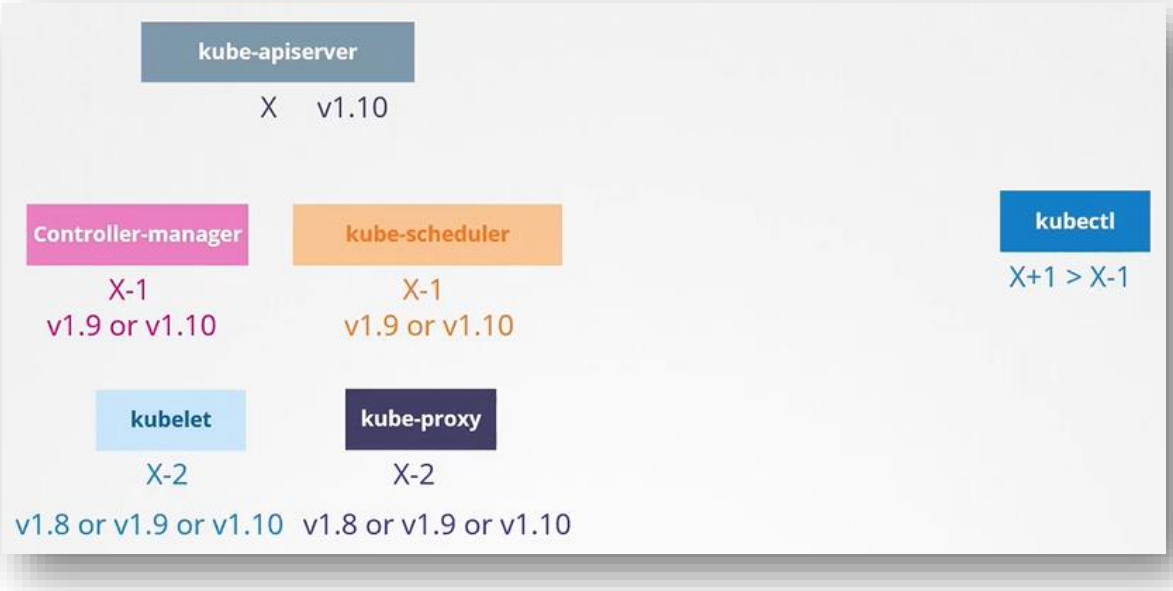
6.3 KUBERNETES SOFTWARE VERSIONS

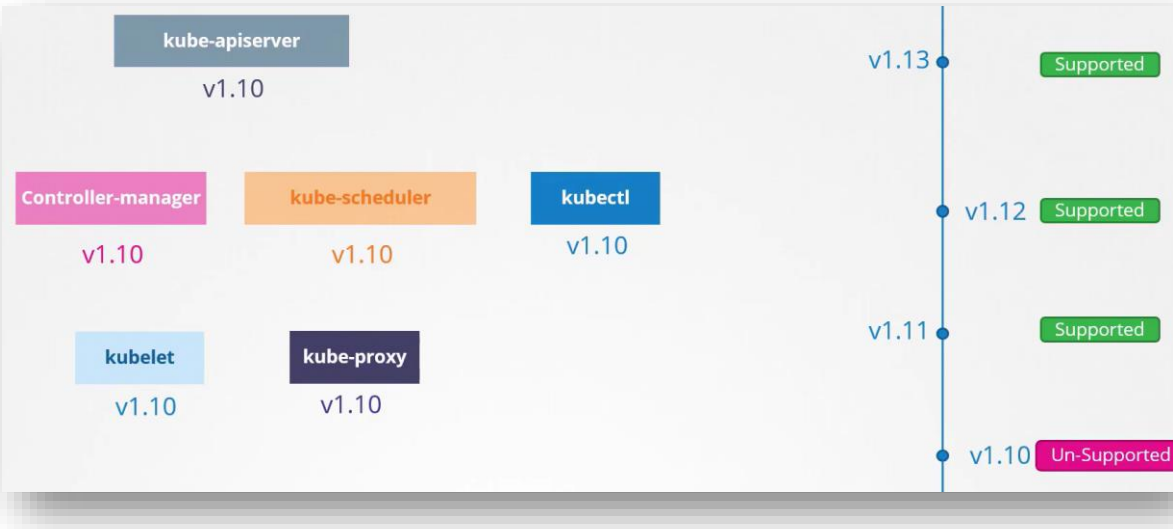

v0.20
June 2015
Mar 2015
v0.12
v0.10
Feb 2015
Jan 2015
v0.8
v0.6
Dec 2014
Oct 2014
v0.4

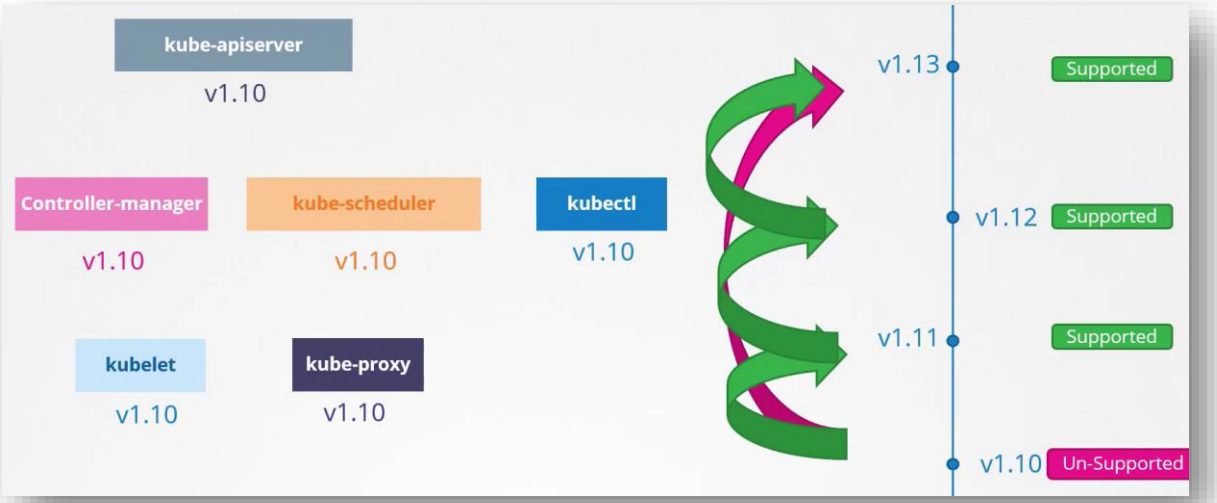

March 2018
V1.10.0
V1.10.0-beta
V1.10.0-alpha

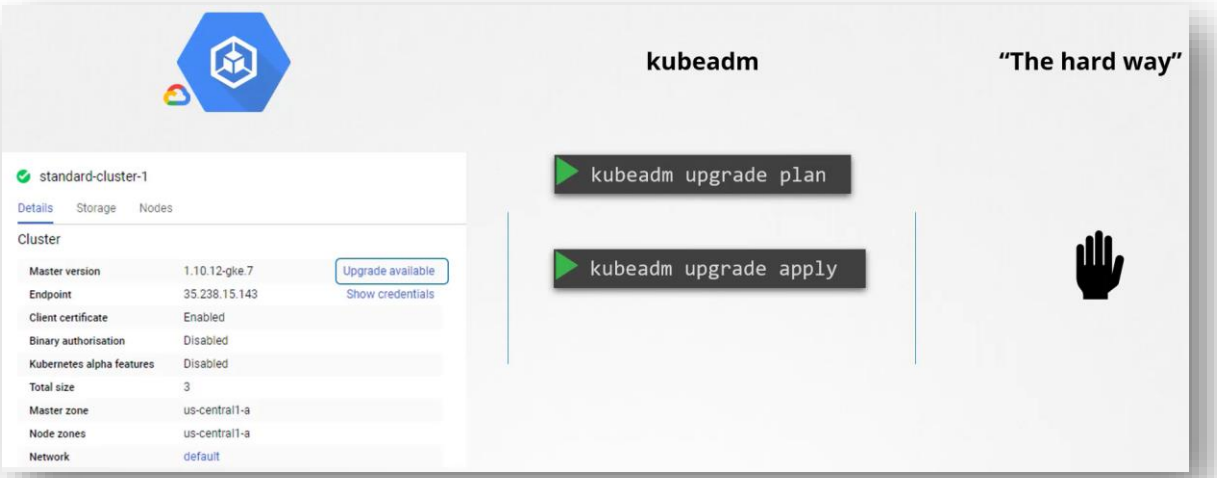

v1.13.4
kube-apiserver
v1.13.4
Controller-manager
v1.13.4
kube-scheduler
v1.13.4
kubelet
v1.13.4
kube-proxy
v1.13.4
kubectl
v1.13.4
ETCD CLUSTER
v3.2.18
CoreDNS
v1.1.3

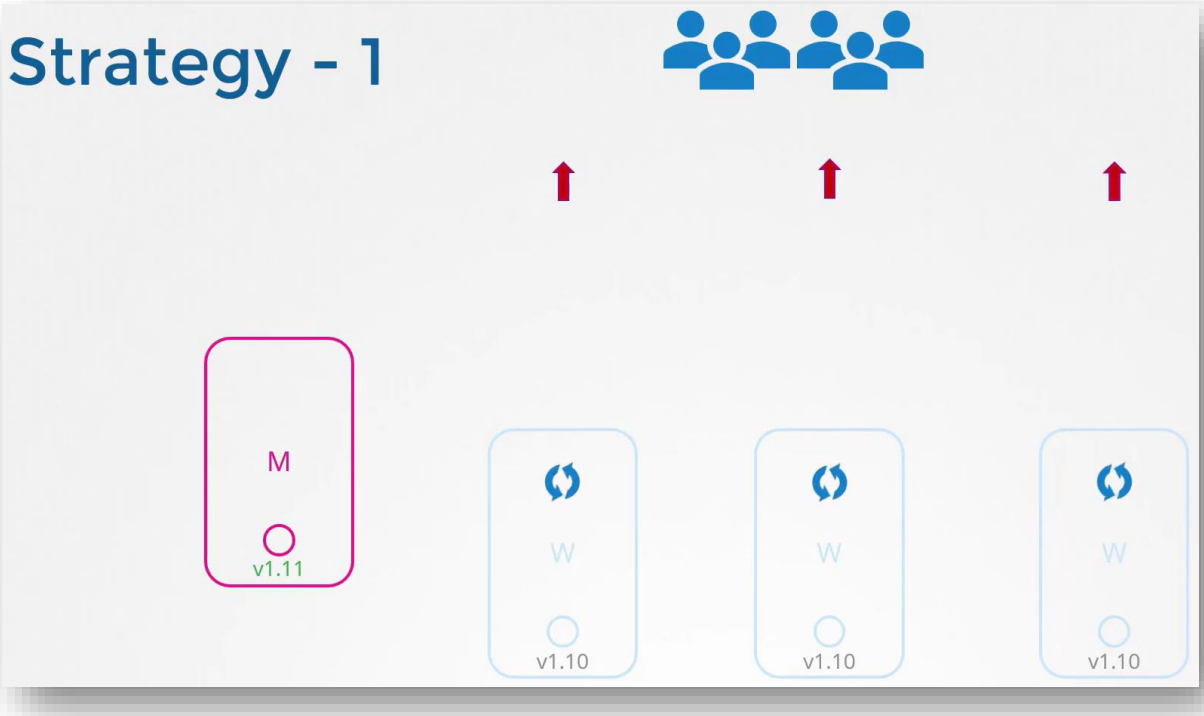
6.4 CLUSTER UPGRADE INTRODUCTION

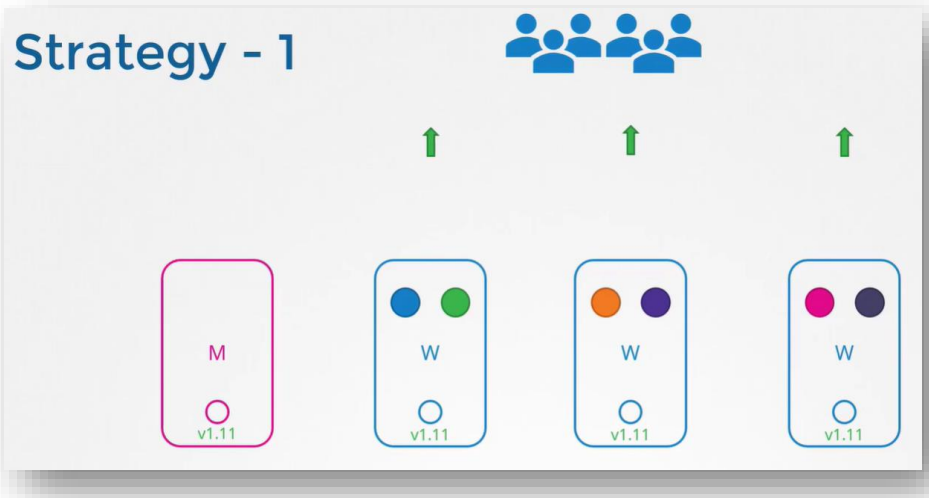


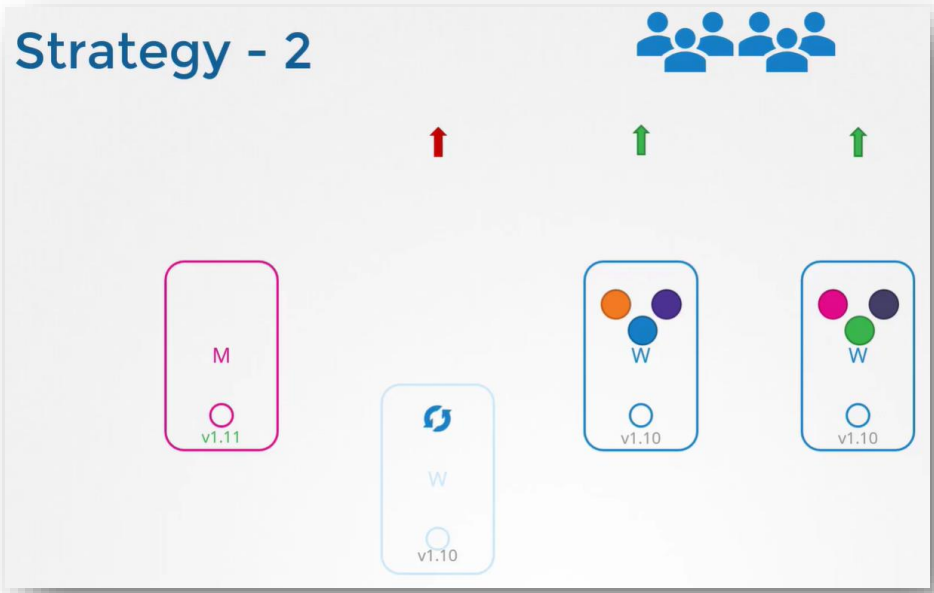


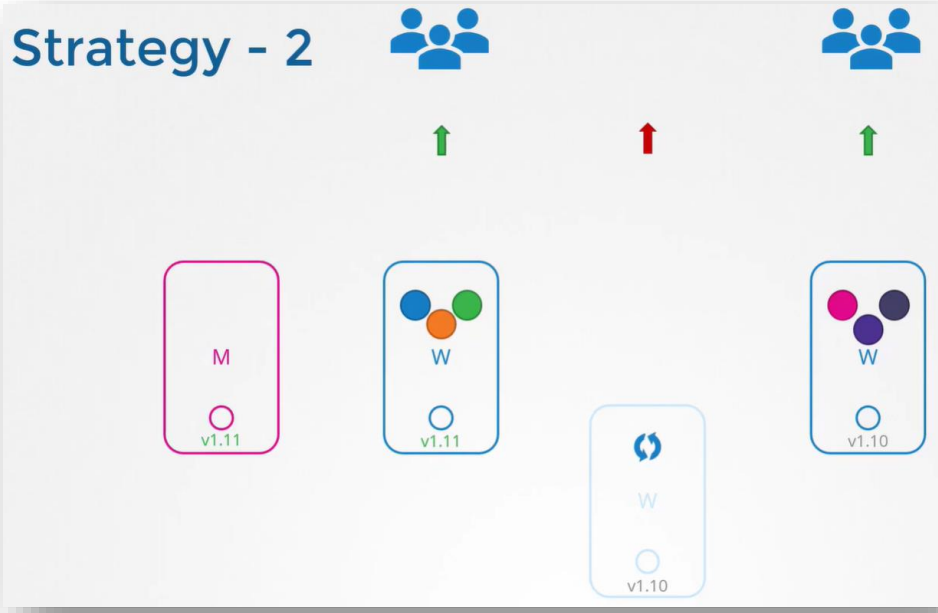

kube-apiserver
v1.10
Controller-manager
v1.10
kube-scheduler
v1.10
kubectl
v1.10
kubelet
v1.10
kube-proxy
v1.10
v1.13
Supported
v1.12
Supported
v1.11
Supported
v1.10
Un-Supported

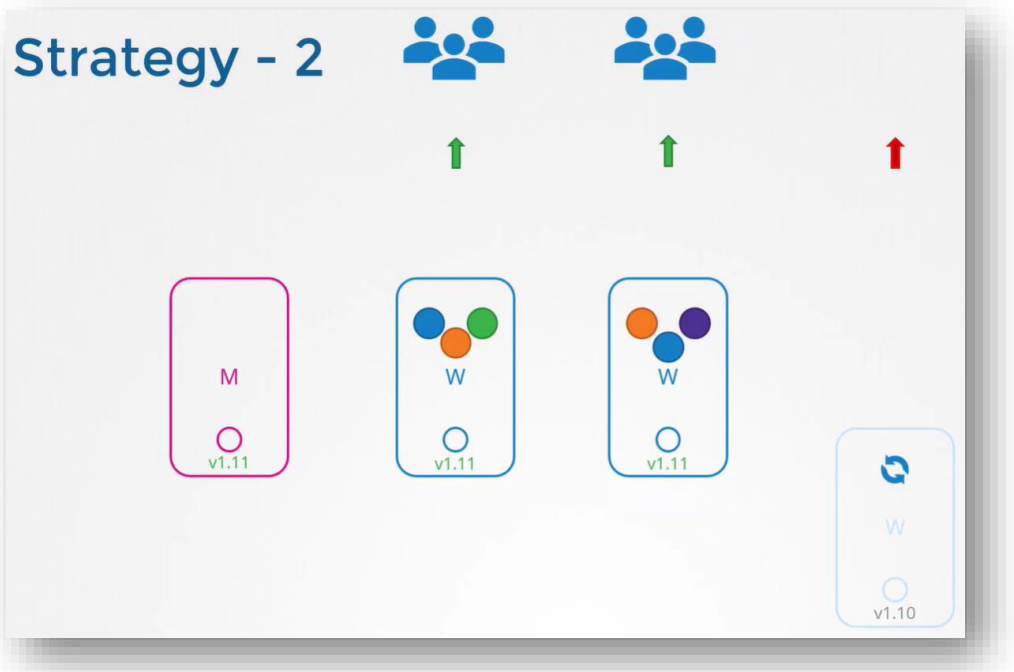

kubeadm
"The hard way"
standard-cluster-1
Details
Storage
Nodes
Cluster
Master version
1.10.12-gke.7
Upgrade available
Endpoint
35.238.15.143
Show credentials
Client certificate
Enabled
Binary authorisation
Disabled
Kubernetes alpha features
Disabled
Total size
3
Master zone
us-central1-a
Node zones
us-central1-a
Network
default
kubeadm upgrade plan
kubeadm upgrade apply

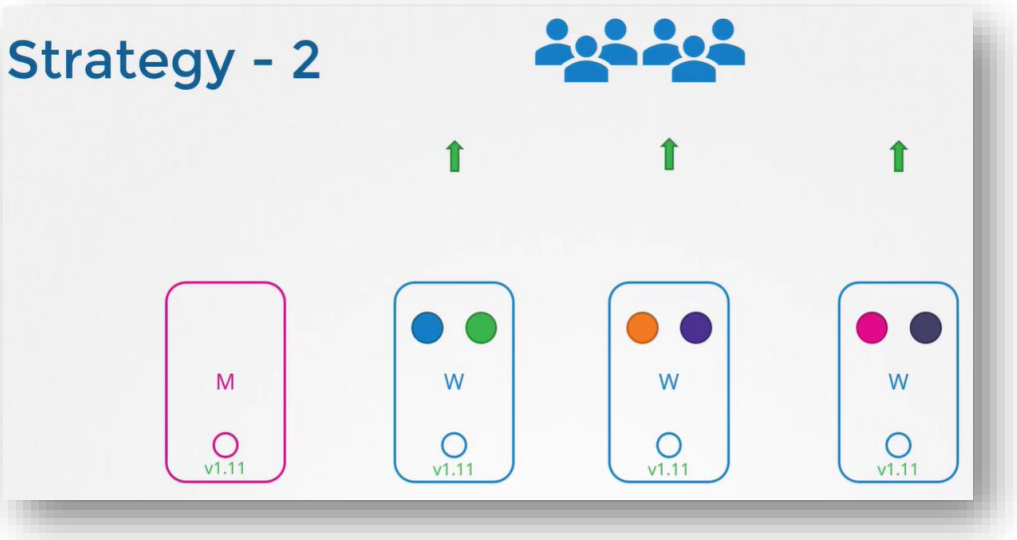

Strategy - 1
M
v1.11
W
v1.10
W
v1.10
W
v1.10

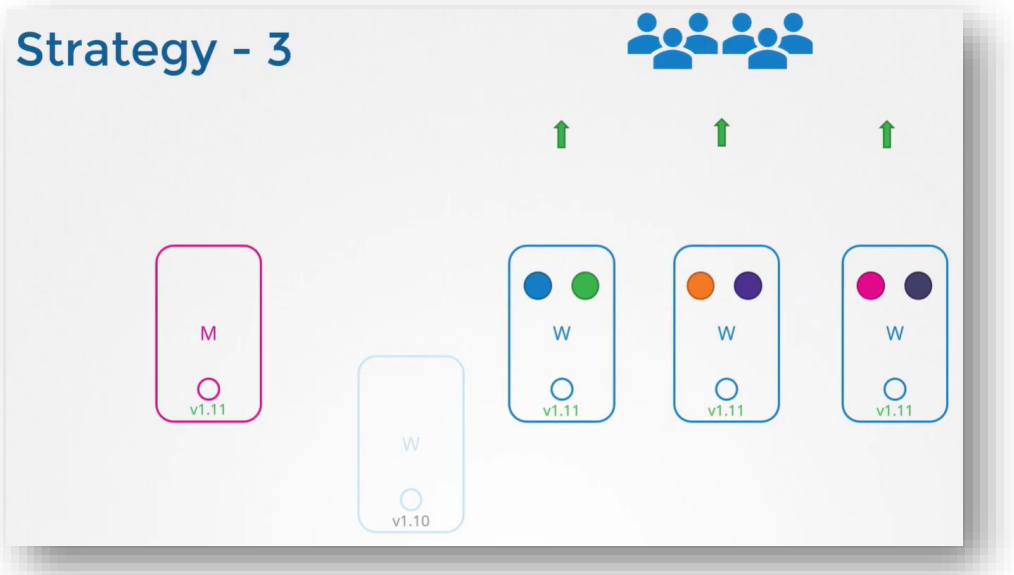

Strategy - 1
M
v1.11
W
v1.11
W
v1.11
W
v1.11

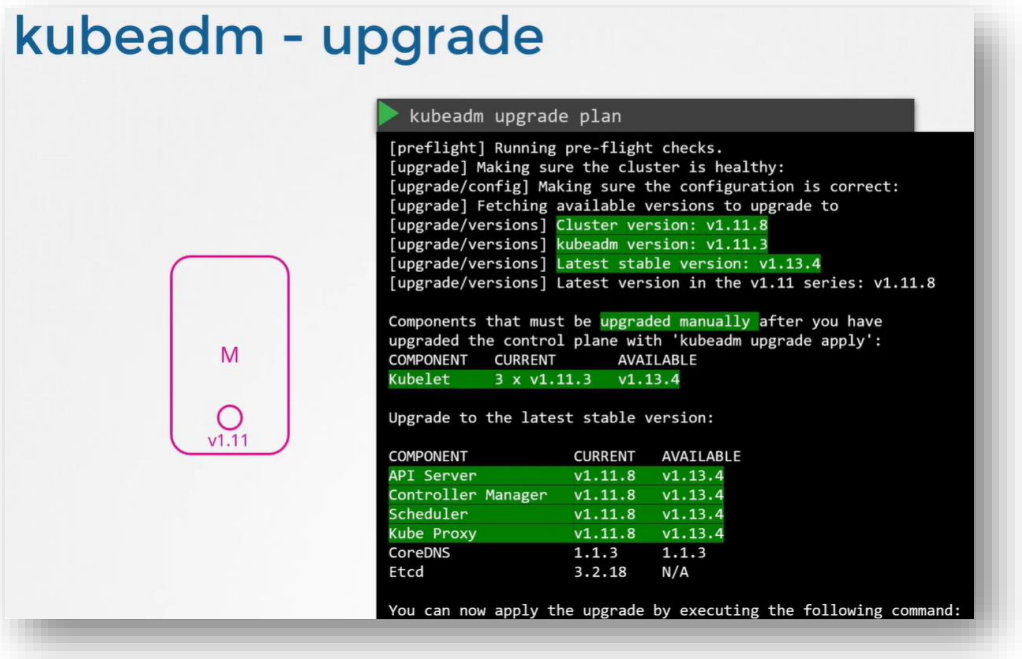

Strategy - 2
M
v1.11
W
v1.10
W
v1.10
W
v1.10


Strategy - 2
M
v1.11
W
v1.11
W
v1.10
W
v1.10

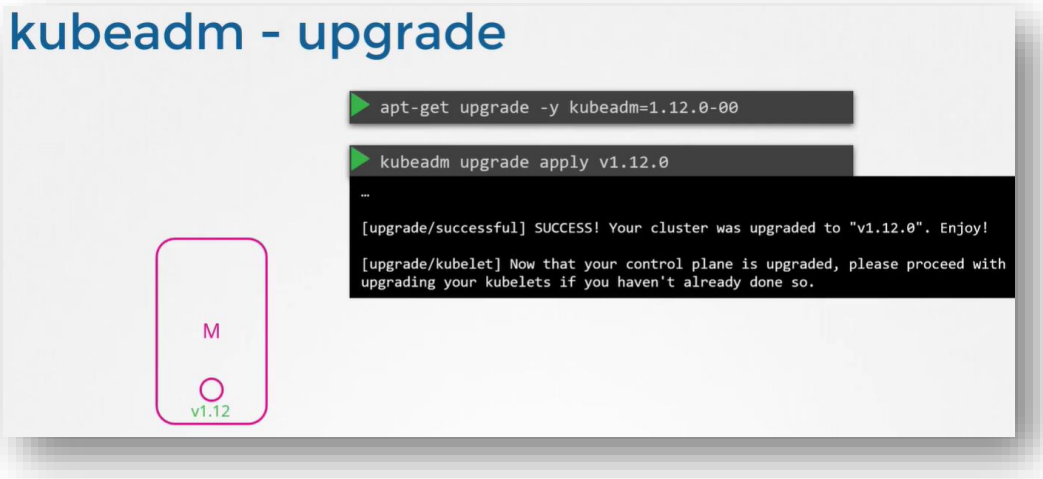

Strategy - 2
M
v1.11
W
v1.11
W
v1.11
W
v1.10

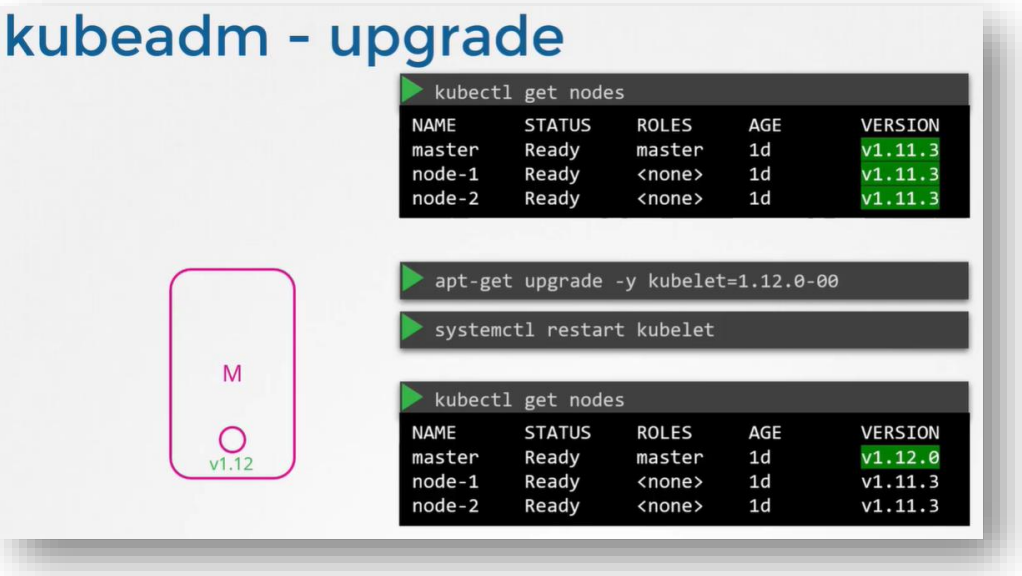

Strategy - 2
M
v1.11
W
v1.11
W
v1.11
W
v1.11

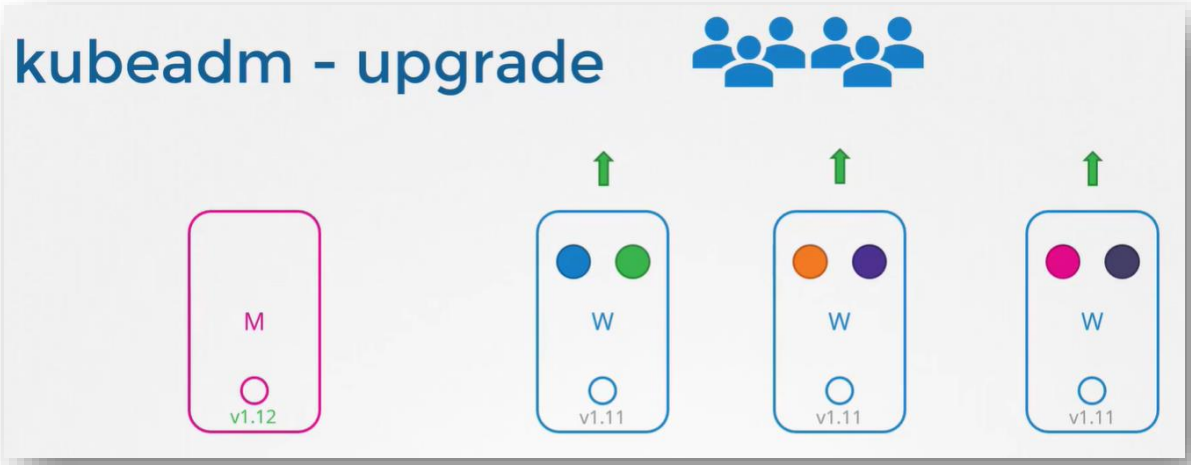

Strategy - 3
M
v1.11
W
v1.10
W
v1.11
W
v1.11
W
v1.11

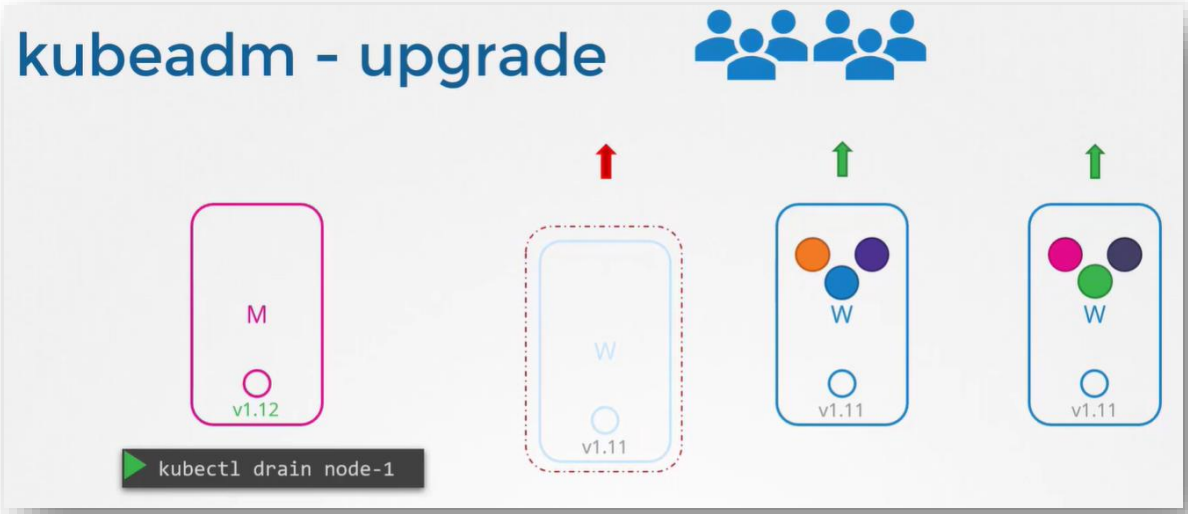


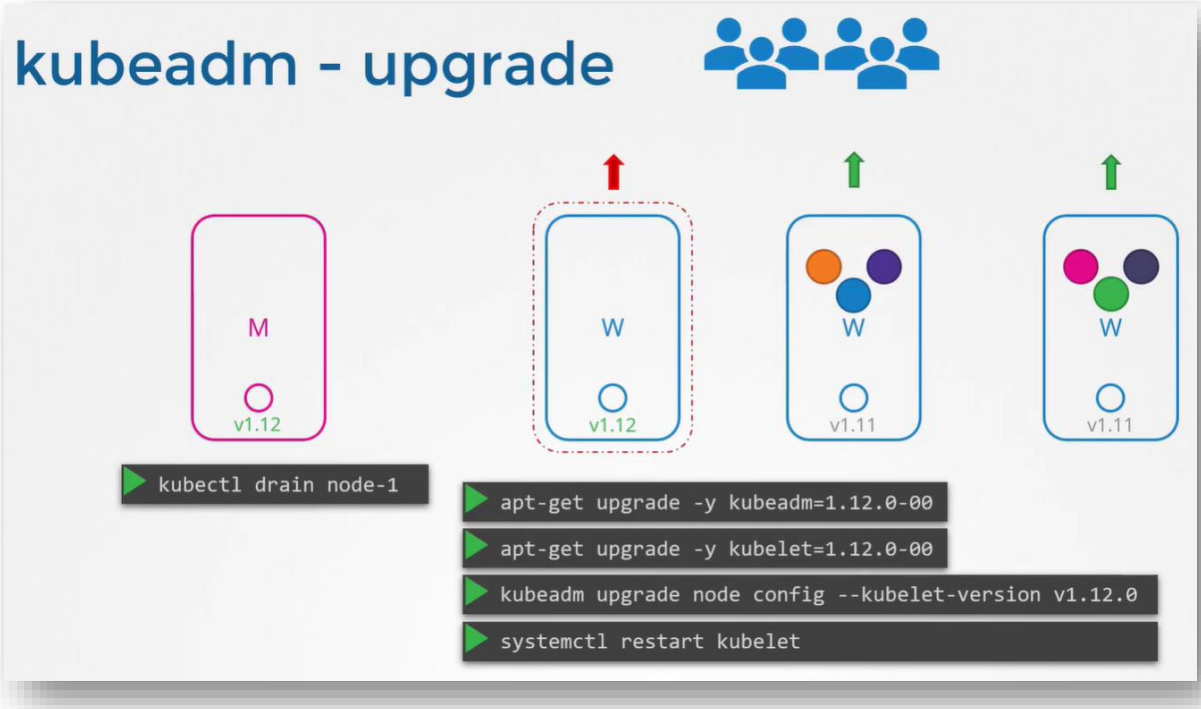
Note: Kubeadm does not install or upgrade kubelet, it will to do upgrade manually.

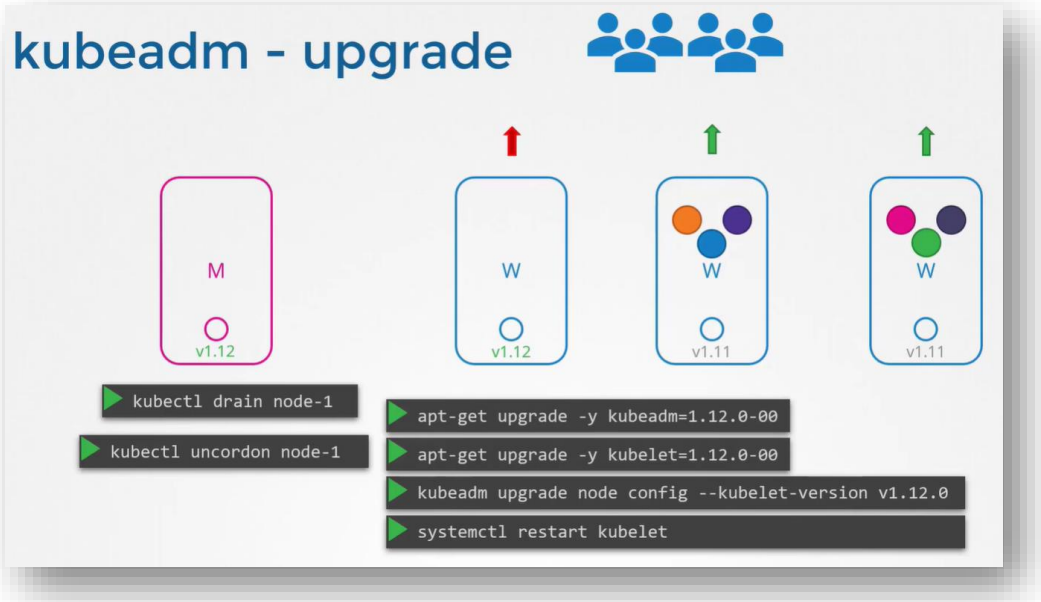


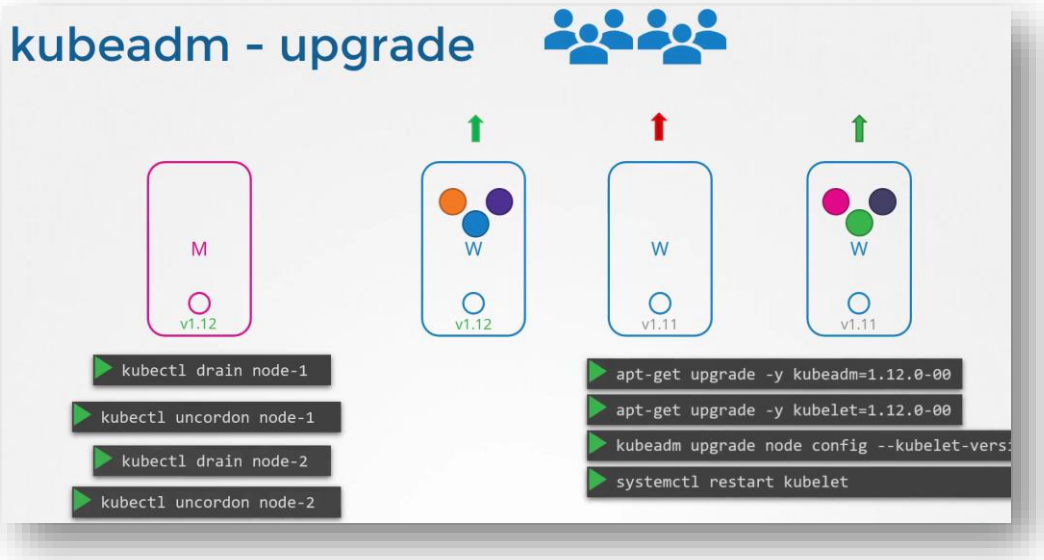


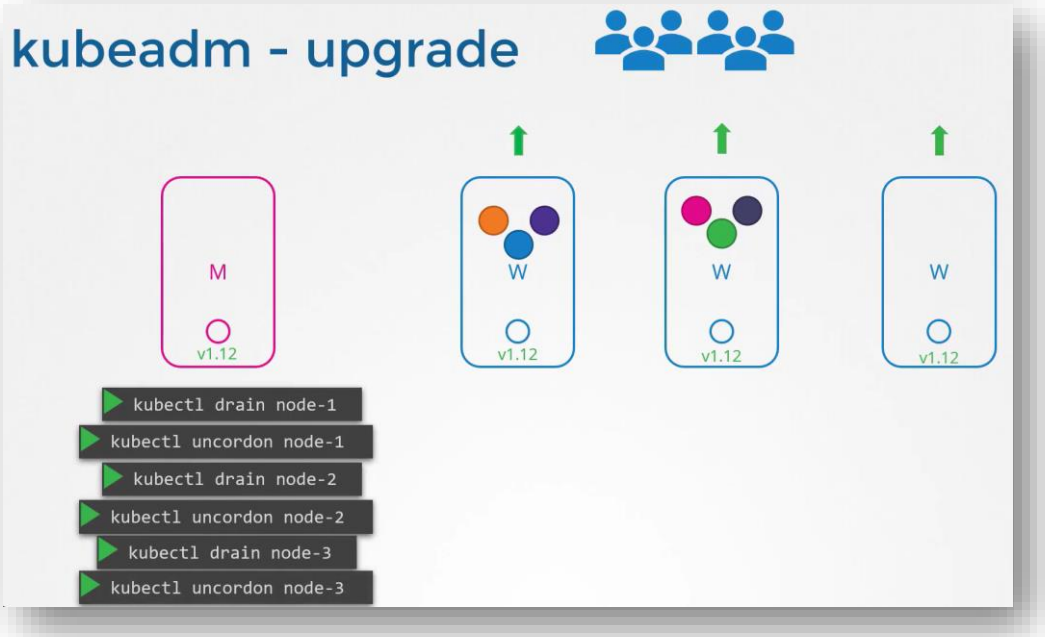

kubeadm - upgrade
M
v1.12
W
v1.11
W
v1.11
W
v1.11


kubeadm - upgrade
M
v1.12
W
v1.11
W
v1.11
W
v1.11
kubectl drain node-1


kubeadm - upgrade
M
v1.12
W
v1.12
W
v1.11
W
v1.11
kubectl drain node-1
apt-get upgrade -y kubeadm=1.12.0-00
apt-get upgrade -y kubelet=1.12.0-00
kubeadm upgrade node config --kubelet-version v1.12.0
systemctl restart kubelet


kubeadm - upgrade
M
v1.12
W
v1.12
W
v1.11
W
v1.11
kubectl drain node-1
kubectl uncordon node-1
apt-get upgrade -y kubeadm=1.12.0-00
apt-get upgrade -y kubelet=1.12.0-00
kubeadm upgrade node config --kubelet-version v1.12.0
systemctl restart kubelet


kubeadm - upgrade
M
v1.12
W
v1.12
W
v1.11
W
v1.11
kubectl drain node-1
kubectl uncordon node-1
kubectl drain node-2
kubectl uncordon node-2
apt-get upgrade -y kubeadm=1.12.0-00
apt-get upgrade -y kubelet=1.12.0-00
kubeadm upgrade node config --kubelet-vers
systemctl restart kubelet


kubeadm - upgrade
M
v1.12
W
v1.12
W
v1.12
W
v1.12
kubectl drain node-1
kubectl uncordon node-1
kubectl drain node-2
kubectl uncordon node-2
kubectl drain node-3
kubectl uncordon node-3

6.4.1 COMMANDLINE

- kubectl drain node01 (Move pod on other nodes).
- kubectl cordon node01 (Mark a node unscheduled)
- kubectl uncordon node01 (Mark a node Schedule)
- kubectl drain node01 --ignore-daemonsets

6.5 DEMO CLUSTER UPGRADE

6.5.1 COMMANDLINE

MasterNode

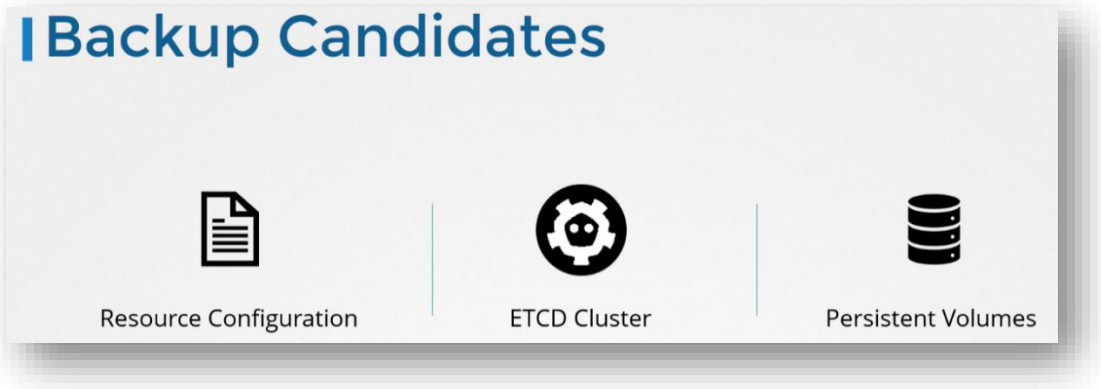
- cat /etc/*release*
- vim /etc/apt/sources.list.d/kubernetes.list
- deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
- apt-get update
- apt-cache madison kubeadm
- apt-get install kubeadm=1.30.0-1.1
- kubeadm version
- kubeadm upgrade plan v1.30.0
- kubeadm upgrade apply v1.30.0
- kubectl get nodes
- kubectl drain controlplane
- kubectl get nodes
- apt-get install kubelet=1.30.0-1.1
- systemctl daemon-reload
- systemctl restart kubelet
- kubectl get nodes
- kubectl uncordon controlplane

WorkerNode

- ssh node01
- cat /etc/*release*
- vim /etc/apt/sources.list.d/kubernetes.list
- deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
- apt-get update
- apt-cache madison kubeadm
- apt-get install kubeadm=1.30.0-1.1
- kubeadm version
- kubeadm upgrade node
- kubectl get nodes
- ssh controlplane
- kubectl drain node01
- ssh node01
- apt-get install kubelet=1.30.0-1.1
- systemctl daemon-reload

- systemctl restart kubelet
- ssh controlplane
- kubectl get nodes
- kubectl uncordon node01
- kubectl get nodes

6.6 BACKUP AND RESTORE METHOD



Declarative

Resource Configuration

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

```
kubectl apply -f pod-definition.yml
```

Backup - ETCD

ETCD Cluster

etcd.service

```
ExecStart=/usr/local/bin/etcd \\
  --name ${ETCD_NAME} \\
  --cert-file=/etc/etcd/kubernetes.pem \\
  --key-file=/etc/etcd/kubernetes-key.pem \\
  --peer-cert-file=/etc/etcd/kubernetes.pem \\
  --peer-key-file=/etc/etcd/kubernetes-key.pem \\
  --trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-client-cert-auth \\
  --client-cert-auth \\
  --initial-advertise-peer-urls https://${INTERNAL_IP}:
  --listen-peer-urls https://${INTERNAL_IP}:2380 \\
  --listen-client-urls https://${INTERNAL_IP}:2379,http
  --advertise-client-urls https://${INTERNAL_IP}:2379 \
  --initial-cluster-token etcd-cluster-0 \\
  --initial-cluster controller-0=https://${CONTROLLER0_
  --initial-cluster-state new \\
  --data-dir=/var/lib/etcd
```

Backup - ETCD

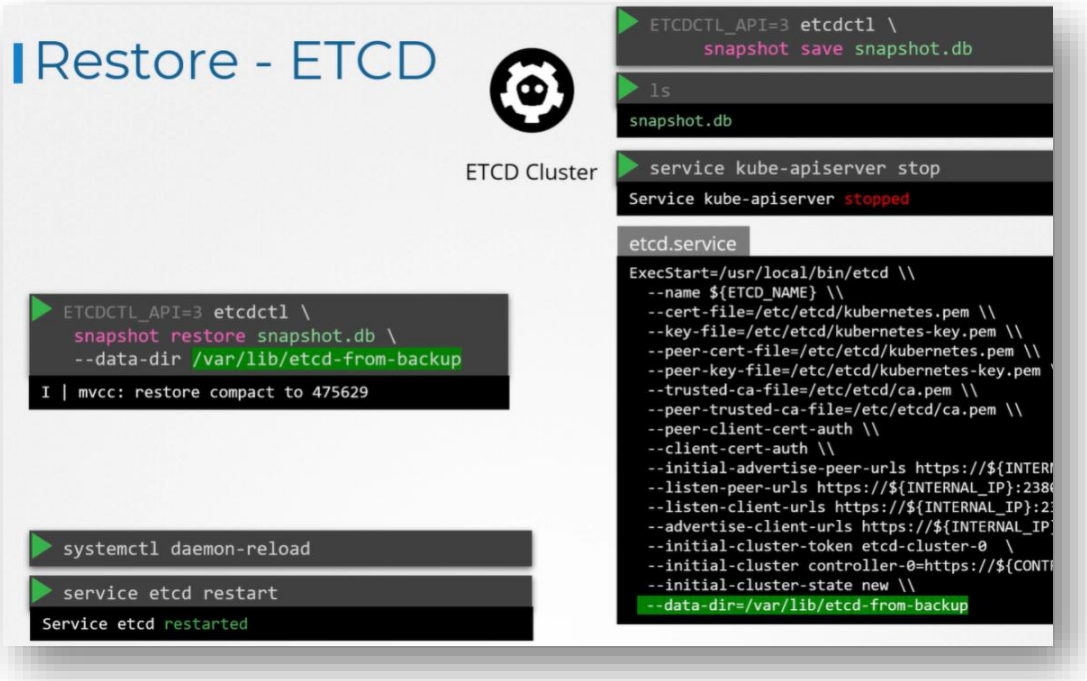
ETCD Cluster

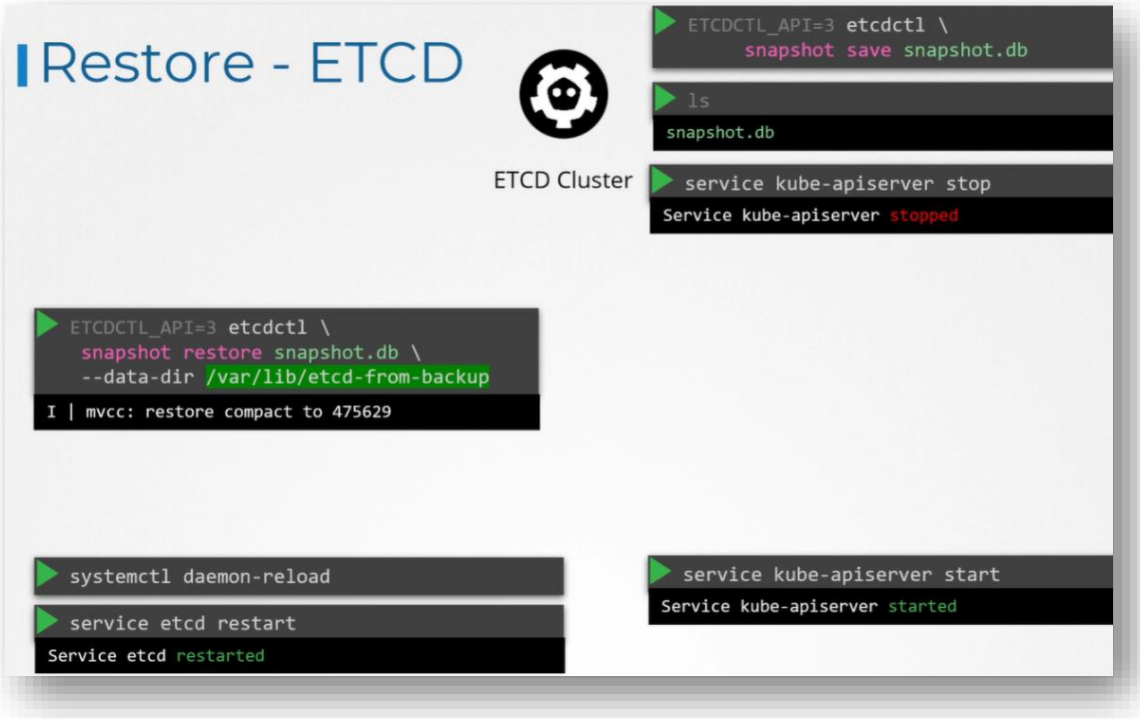

```
ETCDCTL_API=3 etcdctl \
     snapshot save snapshot.db
```

```
ls
snapshot.db
```

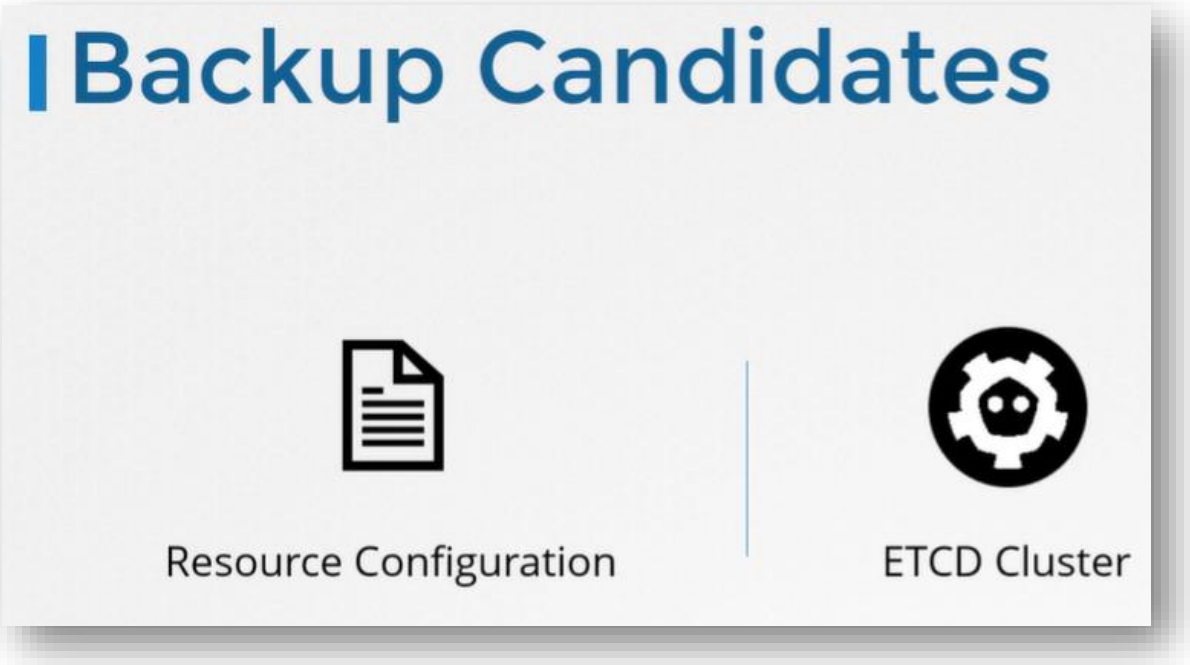
```
ETCDCTL_API=3 etcdctl \
     snapshot status snapshot.db
```

```
+----------+----------+------------+------------+
|   HASH   | REVISION | TOTAL KEYS | TOTAL SIZE |
+----------+----------+------------+------------+
| e63b3fc5 |   473353 |        875 |     4.1 MB |
+----------+----------+------------+------------+
```






```
ETCDCTL_API=3 etcdctl \
     snapshot save snapshot.db \
     --endpoints=https://127.0.0.1:2379 \
     --cacert=/etc/etcd/ca.crt \
     --cert=/etc/etcd/etcd-server.crt \
     --key=/etc/etcd/etcd-server.key
```



6.6.1 COMMANDLINE

- kubectl logs etcd-controlplane -n kube-system
- kubectl -n kube-system logs etcd-controlplane | grep -i 'etcd-version
- kubectl -n kube-system describe pod etcd-controlplane | grep Image:
- kubectl describe pod etcd-controlplane -n kube-system
- kubectl -n kube-system describe pod etcd-controlplane | grep '\--listen-client-urls'
- kubectl -n kube-system describe pod etcd-controlplane | grep '\--cert-file'

6.6.2 ETCD BACKUP AND RESTORE

Backup

- ETCDCTL_API=3 etcd snapshot
- Export ETCDCTL_API=3
- etcdctl snapshot
- etcdctl snapshot save --endpoints=127.0.0.1:2379 \
  > --cacert=/etc/Kubernetes/pki/etcd/ca.crt \
  > --cert=/etc/Kubernetes/pki/etcd/server.crt \
  > --key=/etc/Kubernetes/pki/etcd/server.crt \
  > /opt/snapshot-pre-boot.db
  Snapshot saved at /opt/snapshot-pre-boot.db

Restore

- etcd snapshot restore --data-dir /var/lib/etcd-from-backup /opt/snapshot-pre-boot.db
- ls /var/lib/etcd-from-backup
- check /etc/Kubernetes/manifest/etcd.yaml
- kubectl get pods -n kubesystem
- kubectl get pods -n kube-system –watch
- kubectl logs -n kube-system etcd-controlplane
- kubectl describe pod etcd-controlplane -n kube-system
- kubectl delete pod etcd-controlplane
- kubectl get pods
- kubectl get deploy
- kubectl get svc

6.7 MULTI NODE BACKUP (BACKUP AND RESTORE-2)

6.7.1 COMMANDLINE

- kubectl config view
- kubectl config use-context cluster
- pf -ef | grep -i etcd
- etcdctl member list
- ETCDCTL_API=3 etcdctl --endpoints=127.0.0.1:2379 --cacert=/etc/etcd/pki/ca.pem --cert=/etc/etcd/pki/etcd.pem --key=/etc/etcd/pki/etcd-key.pem member list
-
- ETCDCTL_API=3 etcdctl --endpoints=192.33.162.8:2379 --cacert=/etc/etcd/pki/ca.crt --cert=/etc/etcd/pki/server.cert --key=/etc/etcd/pki/server.key snapshot save /opt/cluster1.db

- ETCDCTL_API=3 etcdctl snapshot restor /root/cluster2.db --data-dir /var/lib/etcd-data-new
- ls /var/lib/etcd-data-new
- chown -R etcd:etcd etcd-data-new/
- ls -la
- vim /etc/systemd/system/etcd.service
- change path
- systemctl daemon reload
- systemctl restart etcd

7.1 SECURITY-SECTION INTRODUCTION

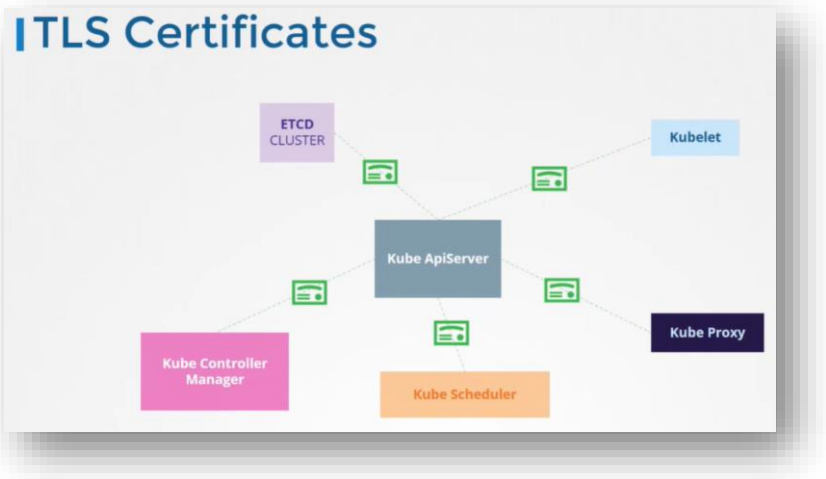
Step 1:

Figure 1 - Authentication types.

Step 2:

Figure 2 - Authentication method.

Step 3:



7.2 KUBERNETES SECURITY PRIMITIVES

7.3 AUTHENTICATION

7.4 ARTICLE ON SETTING UP BASIC AUTHENTICATION

7.5 TLS INTRODUCTION

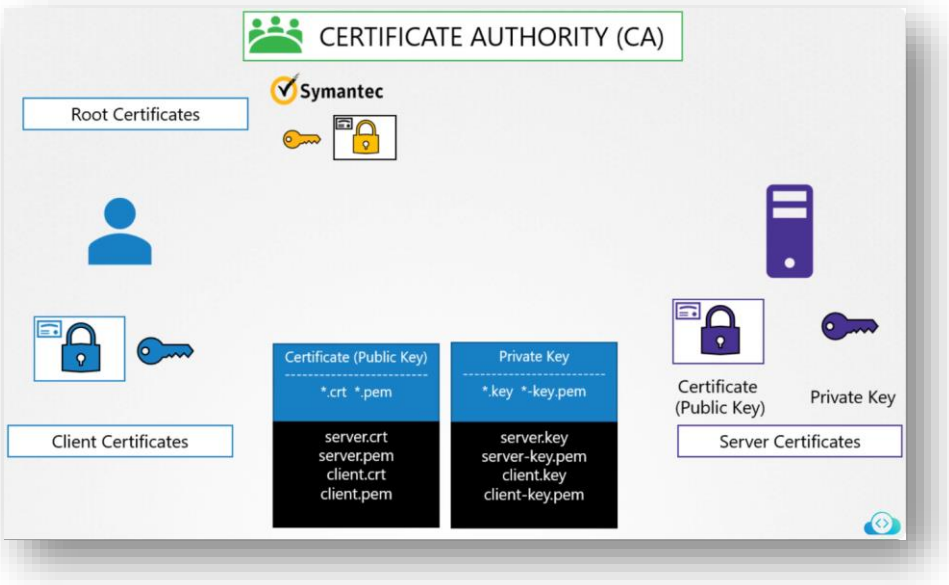
7.6 TLS BASICS

- SYMENTRIC
- ASYMENTRIC

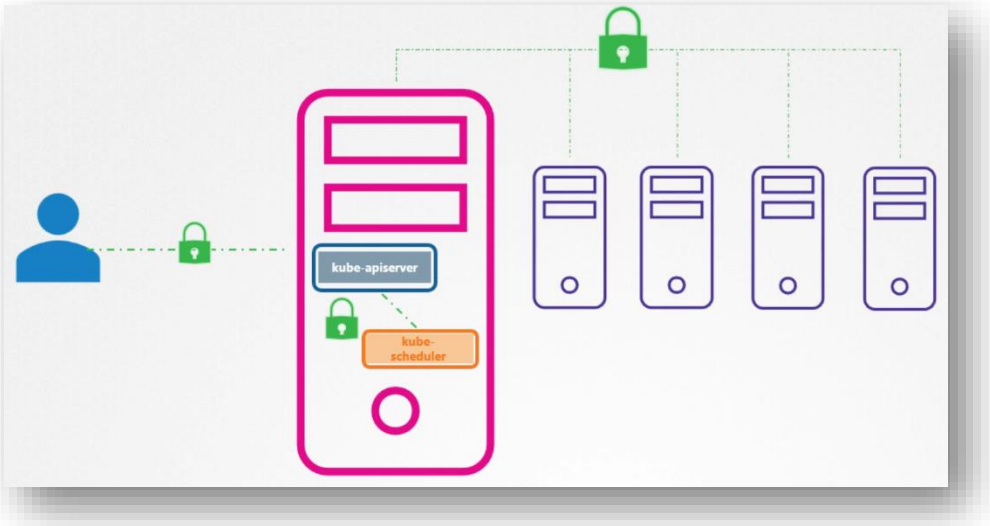
7.7 TLS IN KUBERNETES

- Server Certificate
- Root Certificates
- Client Certificates

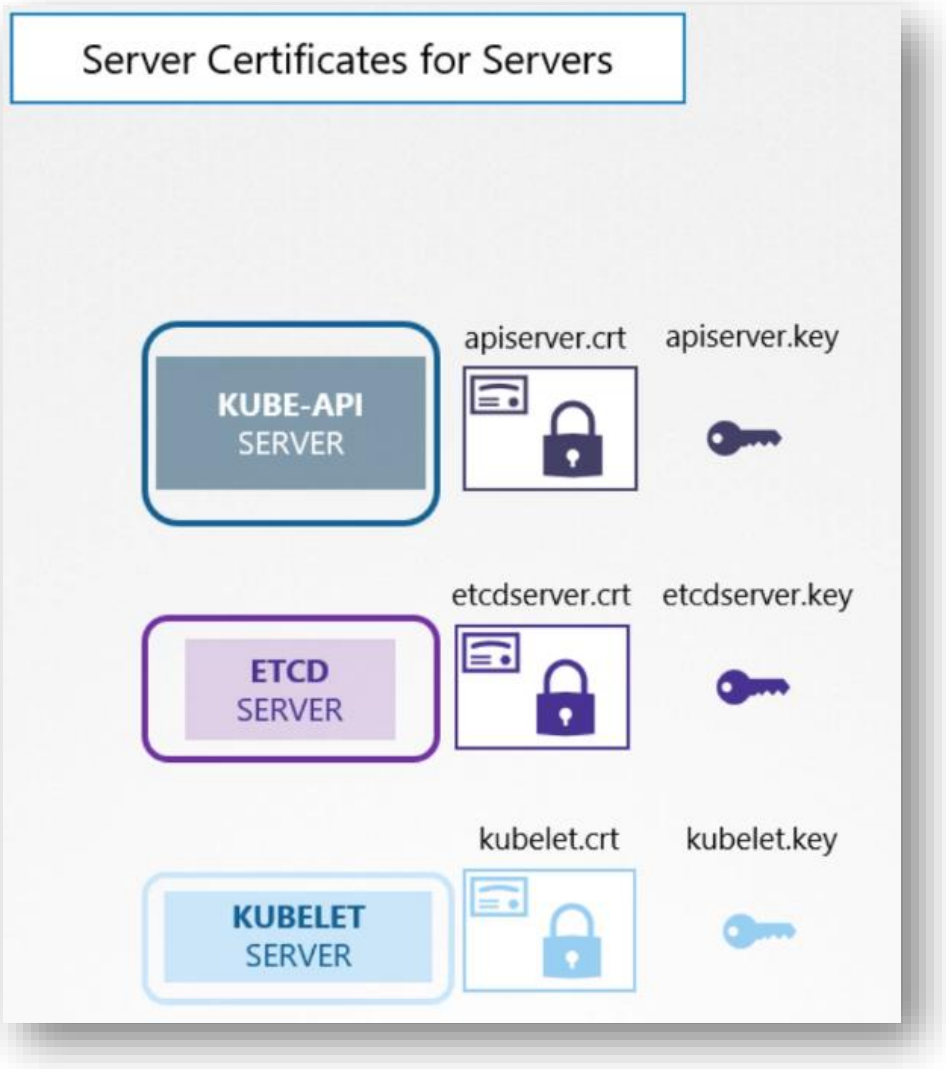
Step 1.


CERTIFICATE AUTHORITY (CA)
Root Certificates
Symantec
Client Certificates
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
Certificate
(Public Key)
Private Key
Server Certificates

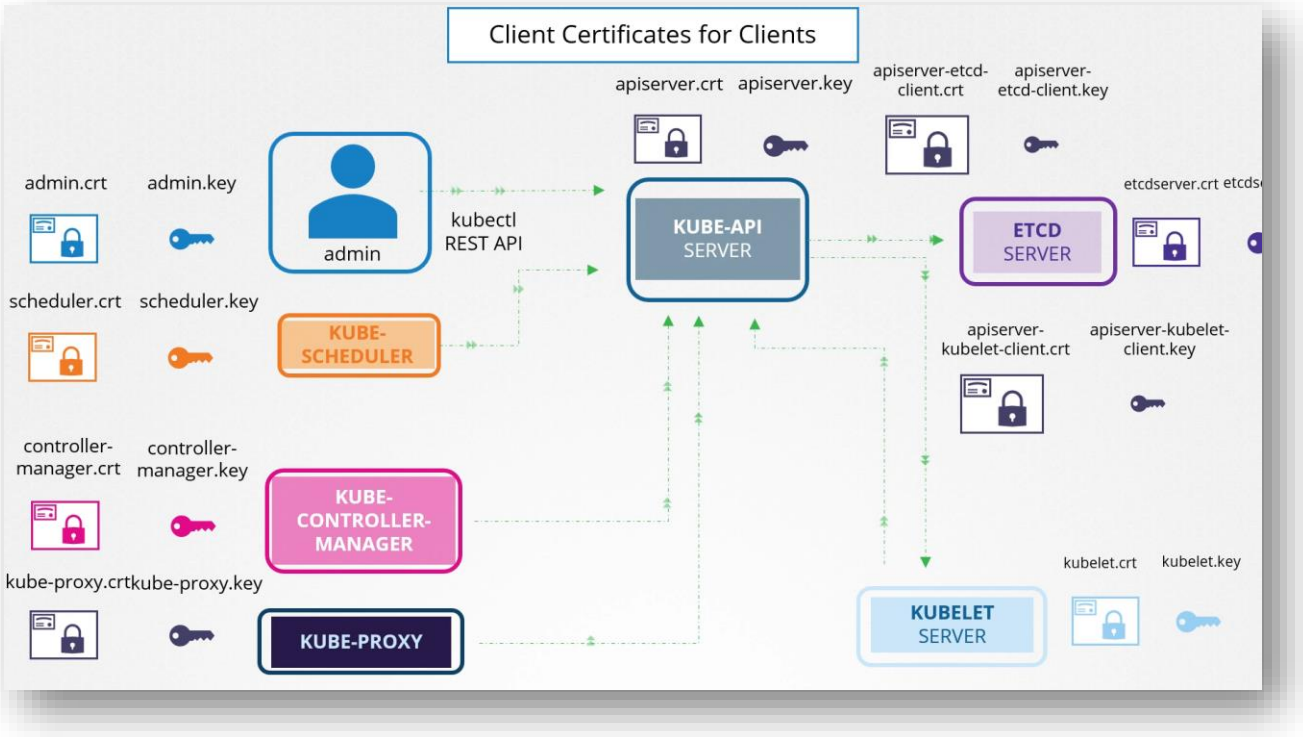
Step 4:


kube-apiserver
kube-
scheduler

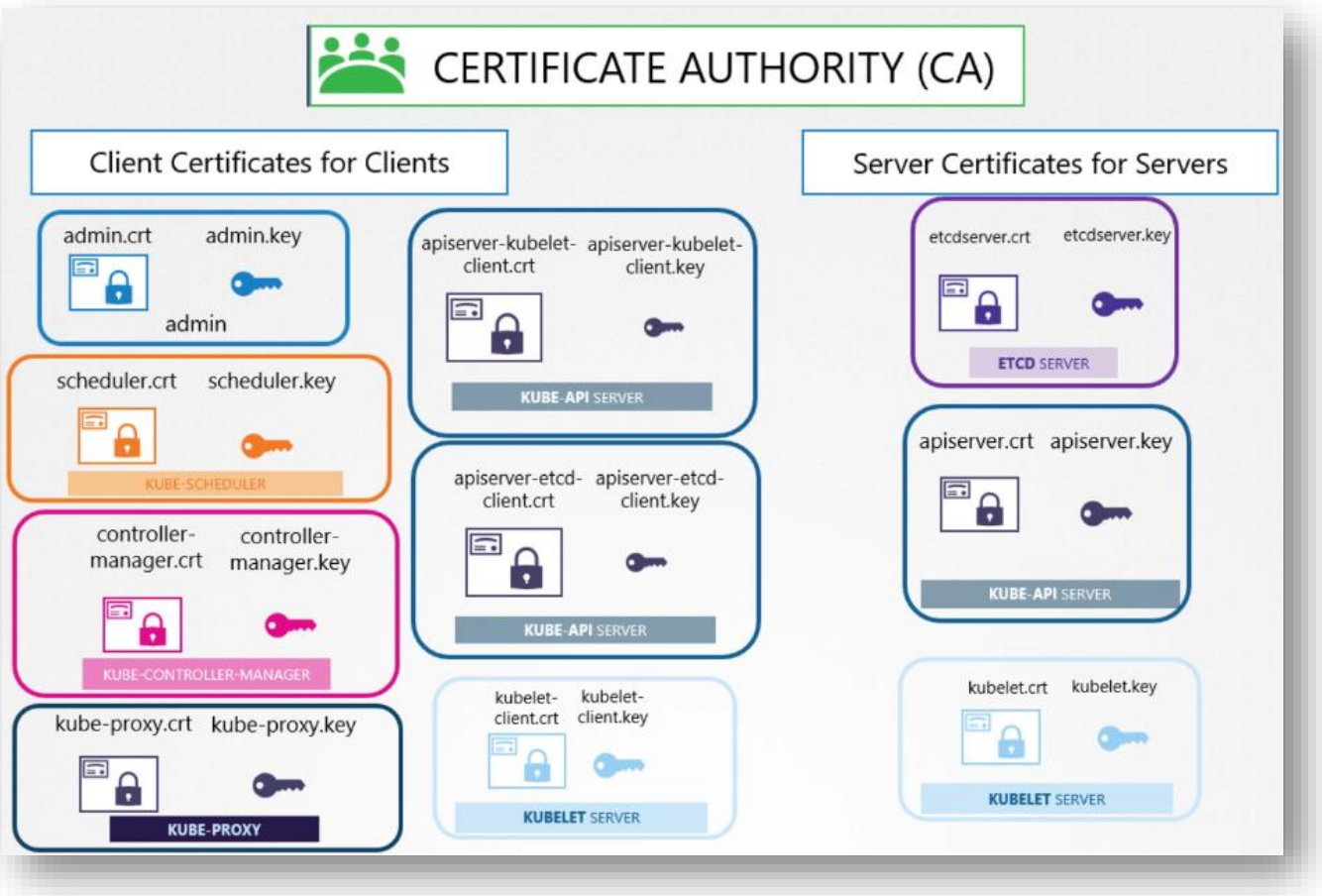
Step 5:


Server Certificates for Servers
KUBE-API SERVER
apiserver.crt
apiserver.key
ETCD SERVER
etcdserver.crt
etcdserver.key
KUBELET SERVER
kubelet.crt
kubelet.key

Step 2.


Client Certificates for Clients
admin.crt
admin.key
admin
kubectl REST API
apiserver.crt
apiserver.key
apiserver-etcd-client.crt
apiserver-etcd-client.key
KUBE-API SERVER
ETCD SERVER
etcdserver.crt
scheduler.crt
scheduler.key
KUBE-SCHEDULER
apiserver-kubelet-client.crt
apiserver-kubelet-client.key
controller-manager.crt
controller-manager.key
KUBE-CONTROLLER-MANAGER
kube-proxy.crt
kube-proxy.key
KUBE-PROXY
kubelet.crt
kubelet.key
KUBELET SERVER

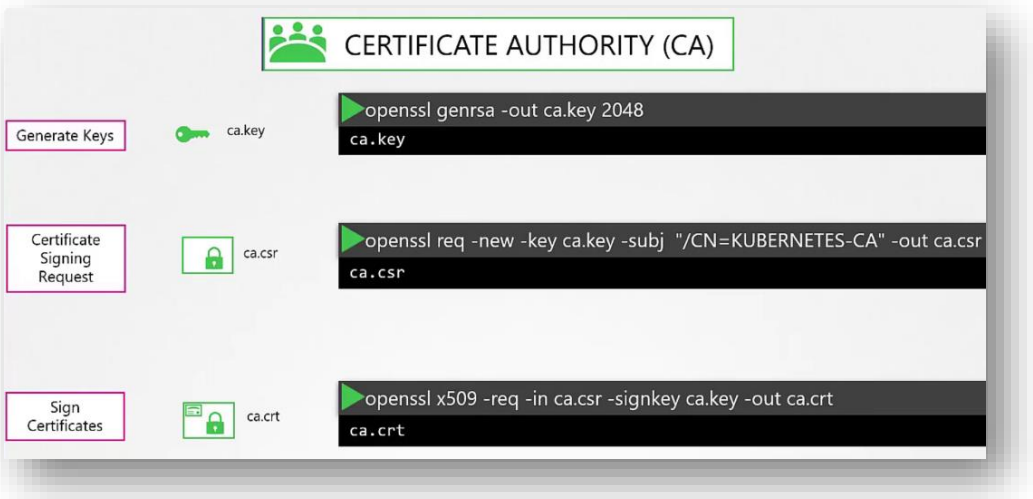
Step 3.



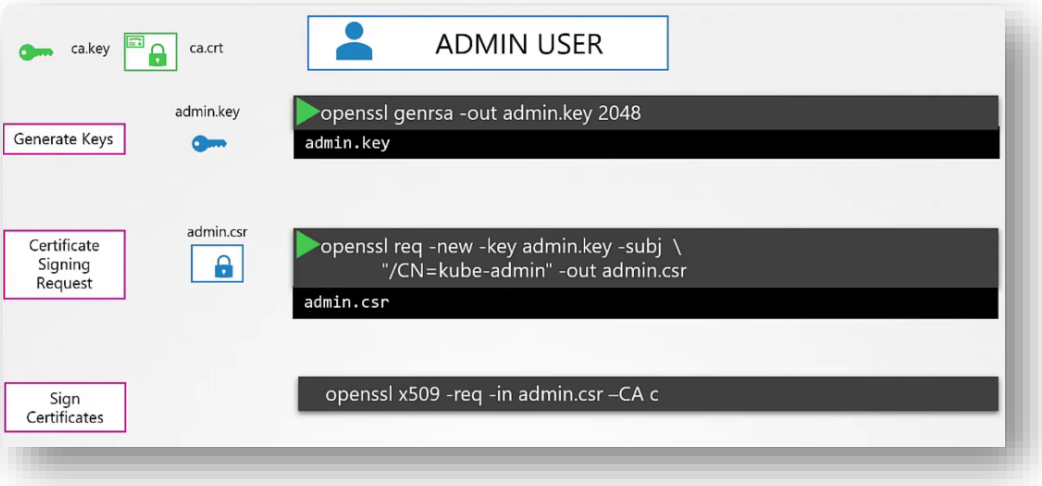
7.8 TLS IN KUBERNETES-CERTIFICATION CREATION

Step 1.

Step 2.


CERTIFICATE AUTHORITY (CA)
Generate Keys
ca.key
openssl genrsa -out ca.key 2048
ca.key
Certificate Signing Request
ca.csr
openssl req -new -key ca.key -subj "/CN=KUBERNETES-CA" -out ca.csr
ca.csr
Sign Certificates
ca.crt
openssl x509 -req -in ca.csr -signkey ca.key -out ca.crt
ca.crt

Step 3.


ca.key
ca.crt
ADMIN USER
admin.key
Generate Keys
openssl genrsa -out admin.key 2048
admin.key
admin.csr
Certificate Signing Request
openssl req -new -key admin.key -subj \
"/CN=kube-admin" -out admin.csr
admin.csr
Sign Certificates
openssl x509 -req -in admin.csr –CA c

Step 4.


ca.key
ca.crt
KUBE SCHEDULER
Generate Keys
scheduler.key
Certificate Signing Request
scheduler.csr
Sign Certificates
scheduler.crt
CERTIFICATE
This Certificate Proudly Presented to
KUBE-SCHEDULER
NEW YORK
NY, US

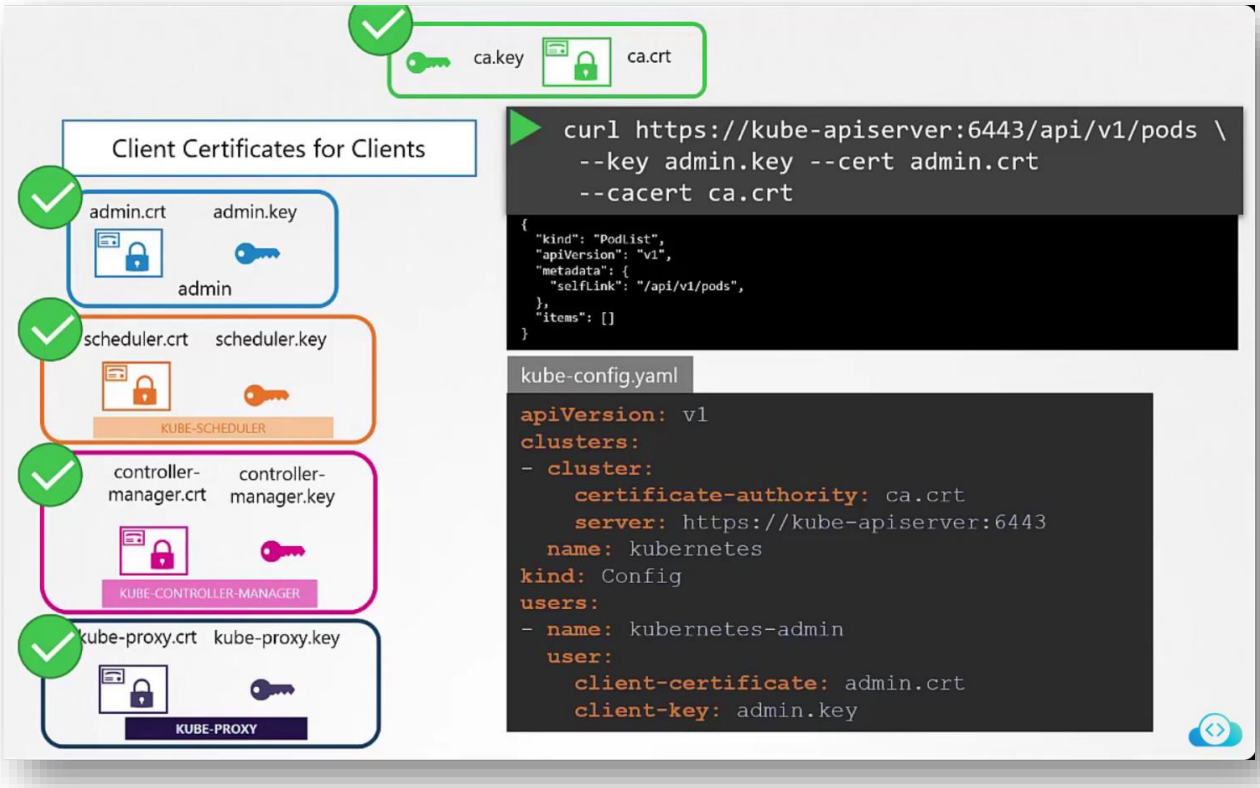
Step 5.


ca.key
ca.crt
KUBE CONTROLLER MANAGER
Generate Keys
controller-manager.key
Certificate Signing Request
controller-manager.csr
Sign Certificates
controller-manager.crt
CERTIFICATE
This Certificate Proudly Presented to
SYSTEMKUBE-CONTROLLER-MANAGER
NEW YORK
NY, US

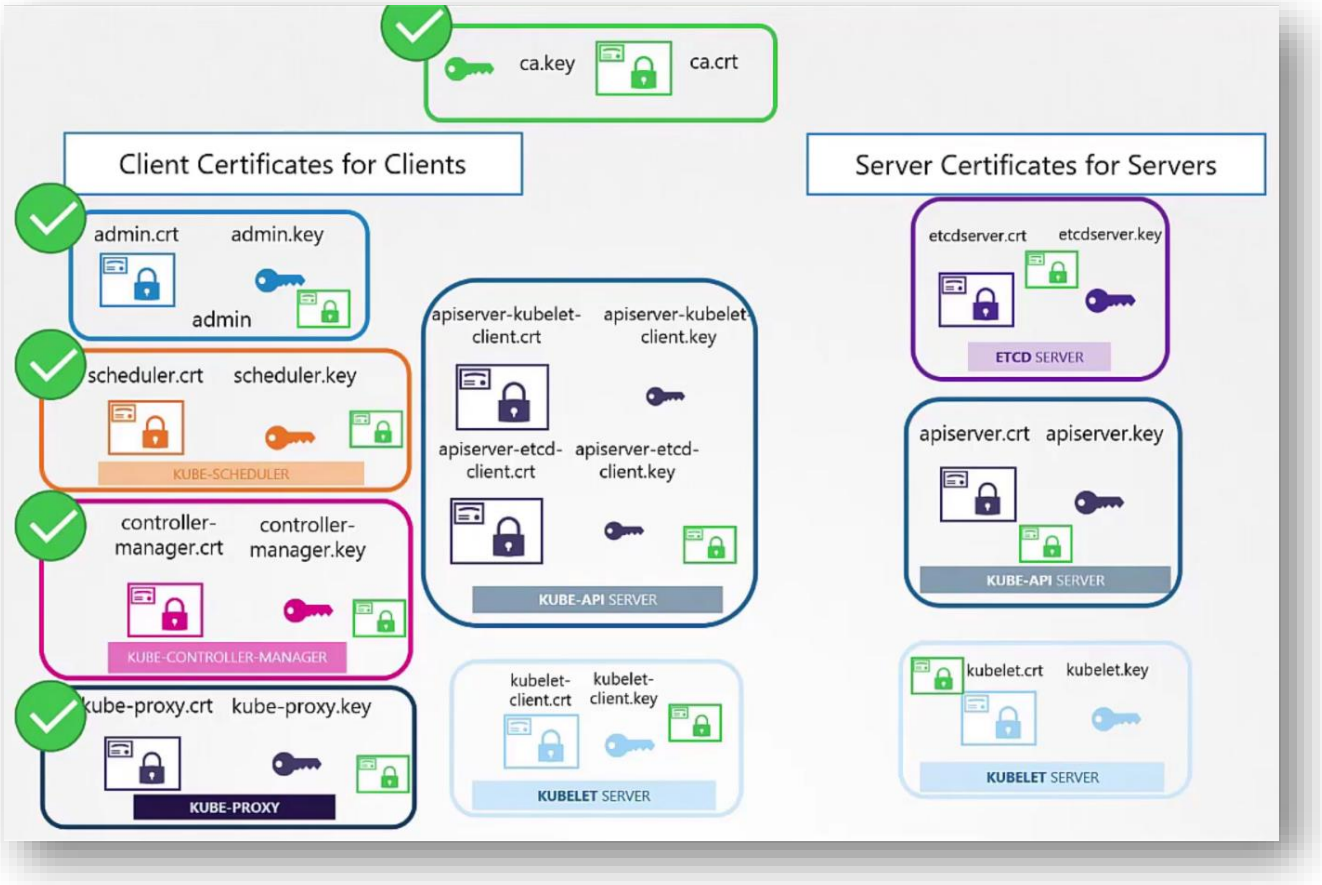
Step 6.


ca.key
ca.crt
KUBE PROXY
Generate Keys
kube-proxy.key
Certificate Signing Request
kube-proxy.csr
Sign Certificates
kube-proxy.crt
CERTIFICATE
This Certificate Proudly Presented to
KUBE-PROXY
NEW YORK
NY, US
Issued by:

Step 7.

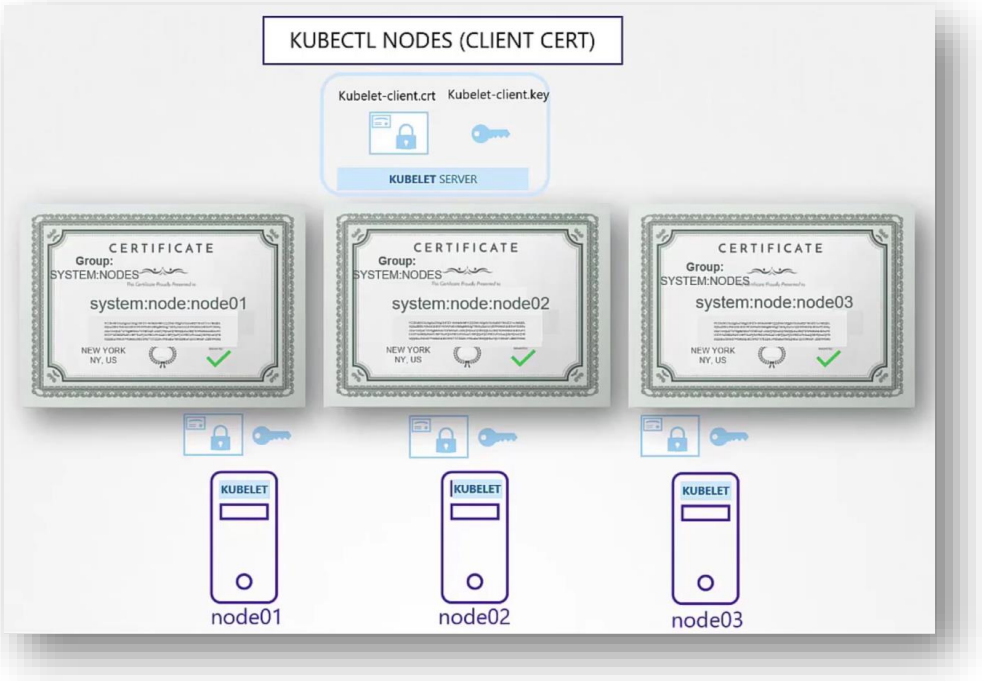

ca.key
ca.crt
Client Certificates for Clients
admin.crt
admin.key
admin
scheduler.crt
scheduler.key
KUBE-SCHEDULER
controller-manager.crt
controller-manager.key
KUBE-CONTROLLER-MANAGER
kube-proxy.crt
kube-proxy.key
KUBE-PROXY
curl https://kube-apiserver:6443/api/v1/pods \
--key admin.key --cert admin.crt
--cacert ca.crt
{
"kind": "PodList",
"apiVersion": "v1",
"metadata": {
"selfLink": "/api/v1/pods",
},
"items": []
}
kube-config.yaml
apiVersion: v1
clusters:
- cluster:
certificate-authority: ca.crt
server: https://kube-apiserver:6443
name: kubernetes
kind: Config
users:
- name: kubernetes-admin
user:
client-certificate: admin.crt
client-key: admin.key

Step 8.



Step 9.

Step 10.

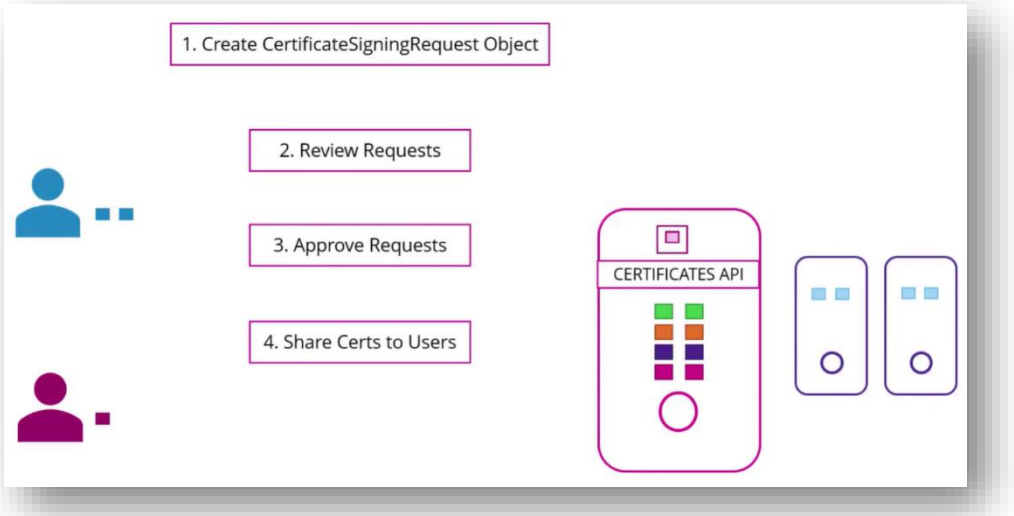


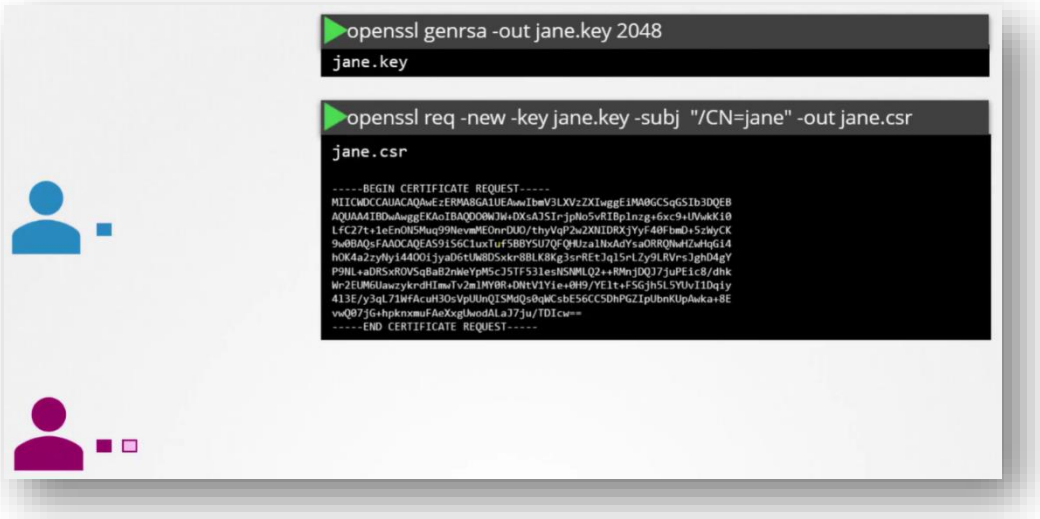
7.8.1 COMMANDLINE

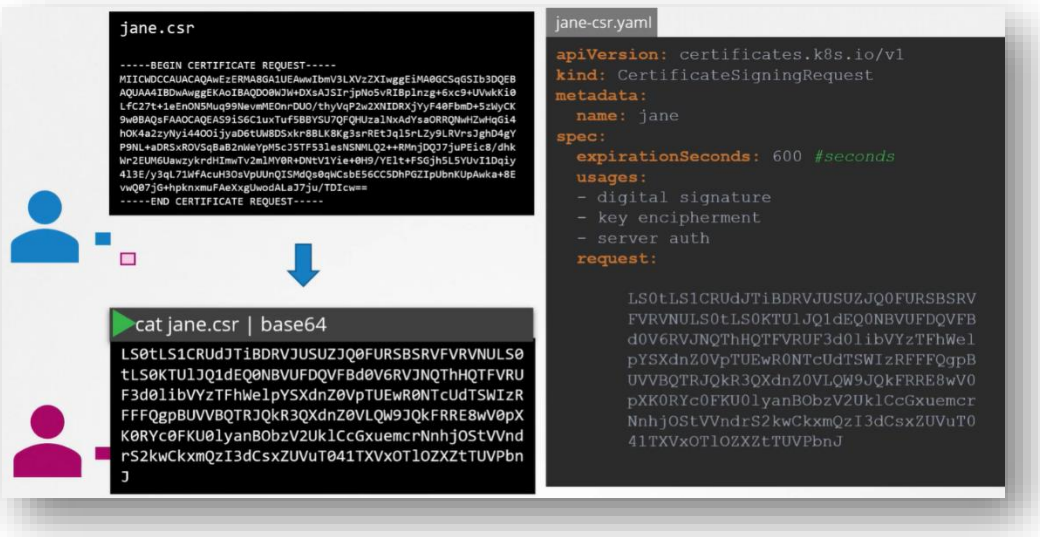
- ls -l /etc/kubernetes/pki/etcd/server* | grep .crt
- crictl ps -a | grep kube-apiserver

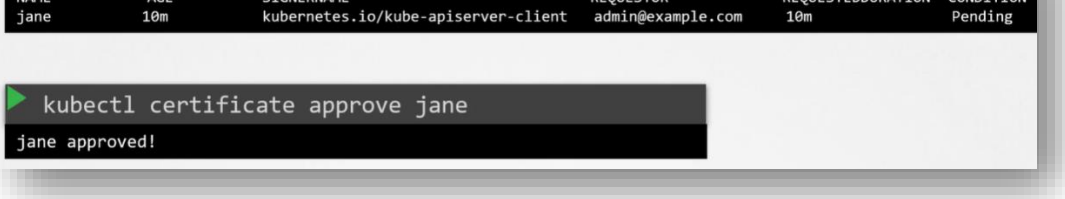
7.9 VIEW CERTIFICATE DETAILS

7.10 CERTIFICATES API




openssl genrsa -out jane.key 2048
jane.key
openssl req -new -key jane.key -subj "/CN=jane" -out jane.csr
jane.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICWDCCAUACAQAwEzERMA8GA1UEAwwIbmV3LXVzZXIwggEiMA0GCSqGSIb3DQEB
AQUAA4IBDwAwggEKAoIBAQDO0WJW+DXsAJSIrjpNo5vRIBplnzg+6xc9+UVwkKi0
LfC27t+1eEnON5Muq99NevmMEOnrDUO/thyVqP2w2XNIDRXjYyF40FbmD+5zWyCK
9w0BAQsFAAOCAQEAS9iS6C1uxTuf5BBYSU7QFQHUzalNxAdYsaORRQNwHZwHqGi4
hOK4a2zyNyi44OOijyaD6tUW8DSxkr8BLK8Kg3srREtJql5rLZy9LRVrsJghD4gY
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF53lesNSNMLQ2++RMnjDQJ7juPEic8/dhk
Wr2EUM6Uawzykrd HImwTv2mlMY0R+DNtV1Yie+0H9/YElt+FSGjh5LSYUvI1Dqiy
413E/y3qL71WfAcuH3OsVpUUnQISMdQs0qWCsbE56CC5DhPGZIpUbnKUpAwka+8E
vwQ07jG+hpknxmuFAeXxgUwodALaJ7ju/TDIcw==
-----END CERTIFICATE REQUEST-----


jane.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICWDCCAUACAQAwEzERMA8GA1UEAwwIbmV3LXVzZXIwggEiMA0GCSqGSIb3DQEB
AQUAA4IBDwAwggEKAoIBAQDO0WJW+DXsAJSIrjpNo5vRIBplnzg+6xc9+UVwkKi0
LfC27t+1eEnON5Muq99NevmMEOnrDUO/thyVqP2w2XNIDRXjYyF40FbmD+5zWyCK
9w0BAQsFAAOCAQEAS9iS6C1uxTuf5BBYSU7QFQHUzalNxAdYsaORRQNwHZwHqGi4
hOK4a2zyNyi44OOijyaD6tUW8DSxkr8BLK8Kg3srREtJql5rLZy9LRVrsJghD4gY
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF53lesNSNMLQ2++RMnjDQJ7juPEic8/dhk
Wr2EUM6Uawzykrd HImwTv2mlMY0R+DNtV1Yie+0H9/YElt+FSGjh5LSYUvI1Dqiy
413E/y3qL71WfAcuH3OsVpUUnQISMdQs0qWCsbE56CC5DhPGZIpUbnKUpAwka+8E
vwQ07jG+hpknxmuFAeXxgUwodALaJ7ju/TDIcw==
-----END CERTIFICATE REQUEST-----
cat jane.csr | base64
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0
tLS0KTUlJQ1dEQ0NBVUFDQVFBd0V6RVJNQThHQTFVRU
F3d0libVYzTFhWelpYSXdnZ0VpTUEwR0NTcUdTSWIzR
FFFQgpBUVVBQTRJQkR3QXdnZ0VLQW9JQkFRRE8wV0pX
K0RYc0FKU0lyanBObzV2UklCcGxuemcrNnhjOStVVnd
rS2kwCkxmQzI3dCsxZUVuT041TXVxOTlOZXZtTUVPbn
J


jane-csr.yaml
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
name: jane
spec:
expirationSeconds: 600 #seconds
usages:
- digital signature
- key encipherment
- server auth
request:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRV
FVRVNULS0tLS0KTUlJQ1dEQ0NBVUFDQVFB
d0V6RVJNQThHQTFVRUF3d0libVYzTFhWel
pYSXdnZ0VpTUEwR0NTcUdTSWIzRFFFQgpB
UVVBQTRJQkR3QXdnZ0VLQW9JQkFRRE8wV0
pXK0RYc0FKU0lyanBObzV2UklCcGxuemcr
NnhjOStVVndrS2kwCkxmQzI3dCsxZUVuT0
41TXVxOTlOZXZtTUVPbnJ

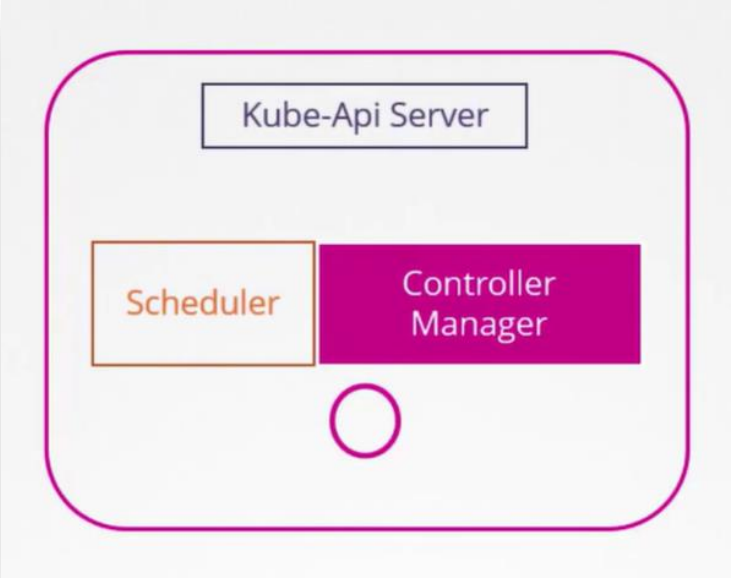
kubectl get csr
NAME AGE SIGNERNAME REQUESTOR REQUESTEDDURATION CONDITION
jane 10m kubernetes.io/kube-apiserver-client admin@example.com 10m Pending
kubectl certificate approve jane
jane approved!

```
kubectl get csr jane -o yaml
```

```
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  creationTimestamp: 2019-02-13T16:36:43Z
  name: new-user
spec:
  groups:
  - system:masters
  - system:authenticated
  expirationSeconds: 600
usages:
  - digital signature
  - key encipherment
  - server auth
  username: kubernetes-admin
status:
  certificate:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURDakNDQWZLZ0F3SUJBZ0lVRmwy
Q2wxYXoxaWl5M3JNVisreFRYQUowU3dnd0RRWUpLb1pJaHZjTkFRRUwKQlFBd0ZURVRN
QkVHQTFVRUF4TUthM1ZpWlhKdVpYUmxjekFlRncweE9UQXlNVE14TmpNeU1EQmFGd1dn
Y0ZFeDl2ajNuSXY3eFdDS1NIRm5sU041c0t5Z0VxUkwzTFM5V29GelhHZDdWCmlEZ2FO
MVVRMFBXTVhjN09FVnVjSWc1Yk4weEVHTkVwRU5tdUlBNlZWeHVjS1h6aG9ldDY0MEd1
MGU0YXFKWVIKWmVMbjBvRTFCY3dod2xic0I1ND0KLS0tLS1FTkQgQ0VSVElGSUNBVEUt
LS0tLQo=
  conditions:
  - lastUpdateTime: 2019-02-13T16:37:21Z
    message: This CSR was approved by kubectl certificate approve.
    reason: KubectlApprove
    type: Approved
```

```
echo "LS0…Qo=" | base64 --decode
```

```
-----BEGIN CERTIFICATE -----
MIICWDCCAUACAQAwEzERMA8GA1UEAwwIbmV3LXVzZXIwgg
AQUAA4IBDwAwggEKAoIBAQDO0WJW+DXsAJSIrjpNo5vRIE
LfC27t+1eEnON5Muq99NevmMEOnrDUO/thyVqP2w2XNIDR
y3BihhB93MJ7Oql3UTvZ8TELqyaDknRl/jv/SxgXkok0AE
IF5nxAttMVkDPQ7NbeZRG43b+QWlVGR/z6DWOfJnbfezOt
EcCXAwqChjBLkz2BHPR4J89D6Xb8k39pu6jpyngV6uP0tI
j2qEL+hZEWkkFz80lNNtyT5LxMqENDCnIgwC4GZiRGbrAg
9w0BAQsFAAOCAQEAS9iS6C1uxTuf5BBYSU7QFQHUzalNxA
hOK4a2zyNyi44OOijyaD6tUW8DSxkr8BLK8Kg3srREtJql
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF53lesNSNMLQ2++R
Wr2EUM6UawzykrdHImwTv2mlMY0R+DNtV1Yie+0H9/YElt
4l3E/y3qL71WfAcuH3OsVpUUnQISMdQs0qWCsbE56CC5DH
vwQ07jG+hpknxmuFAeXxgUwodALaJ7ju/TDIcw==
-----END CERTIFICATE -----
```



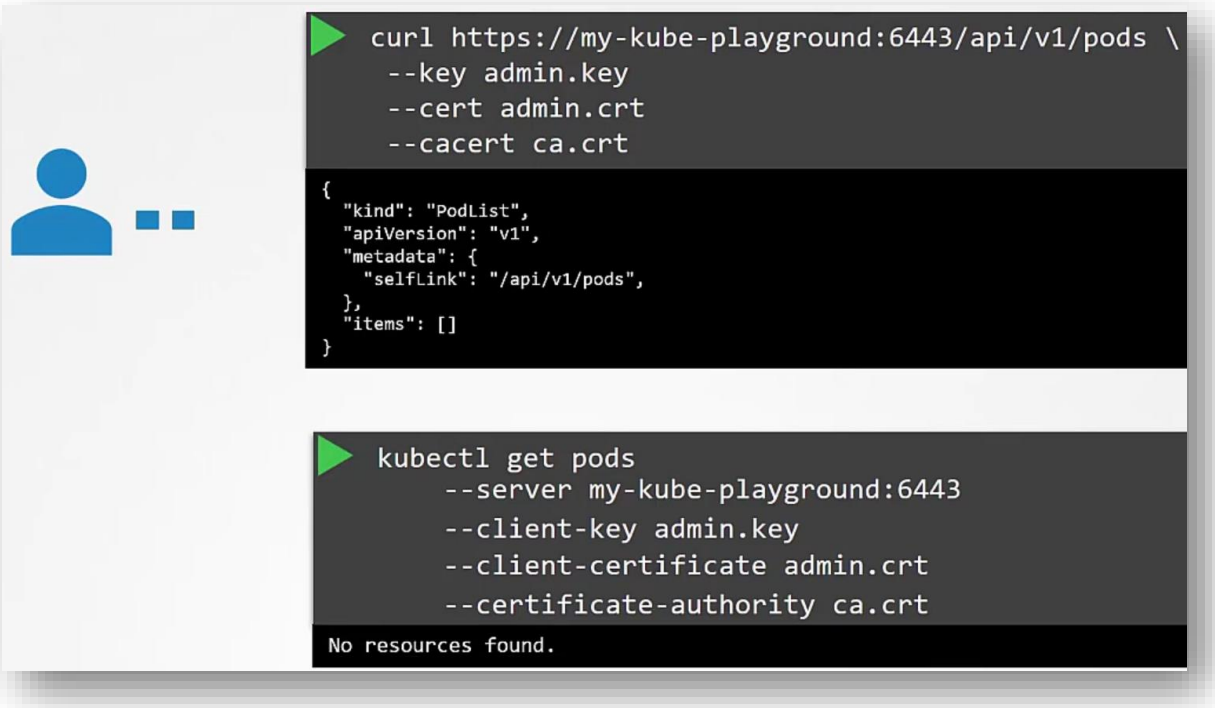
```
cat /etc/kubernetes/manifests/kube-controller-manager.yaml
spec:
  containers:
  - command:
    - kube-controller-manager
    - --address=127.0.0.1
    - --cluster-signing-cert-file=/etc/kubernetes/pki/ca.crt
    - --cluster-signing-key-file=/etc/kubernetes/pki/ca.key
    - --controllers=*,bootstrapsigner,tokencleaner
    - --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
```

7.10.1 COMMANDLINE

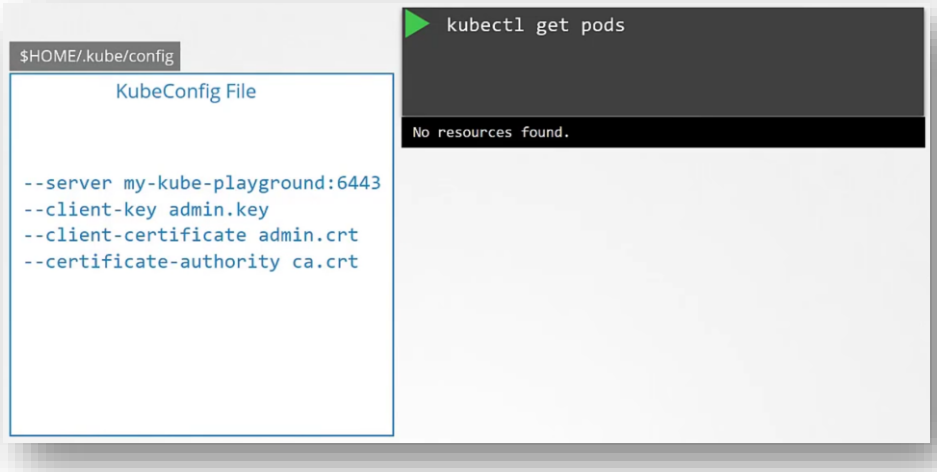
- cat akshay.csr | base64 -w 0
- kubectl certificate approve username
- kubectl get csr
- kubectl get csr agent-smith -o yaml
- kubectl certificate deny agent-smith
- kubectl delete csr agent-smith

7.11 KUBE CONFIG

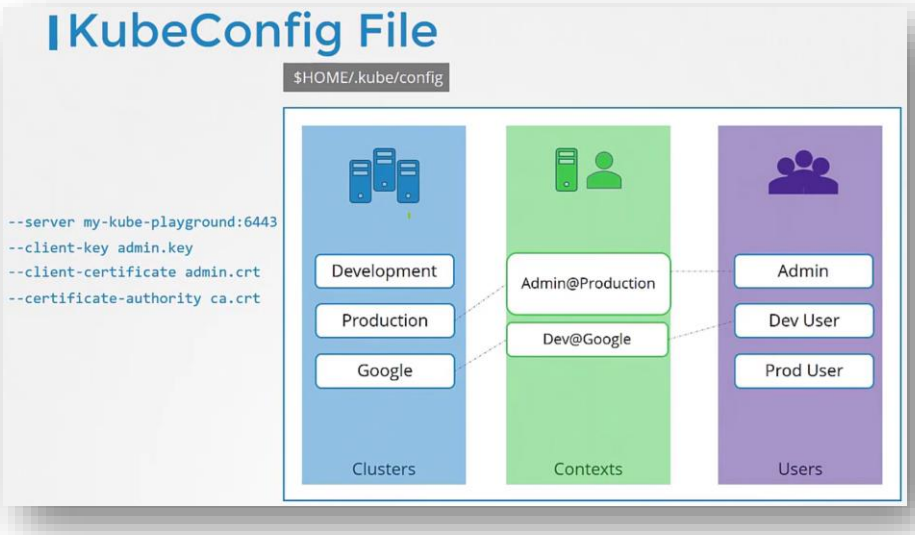
Step 1.



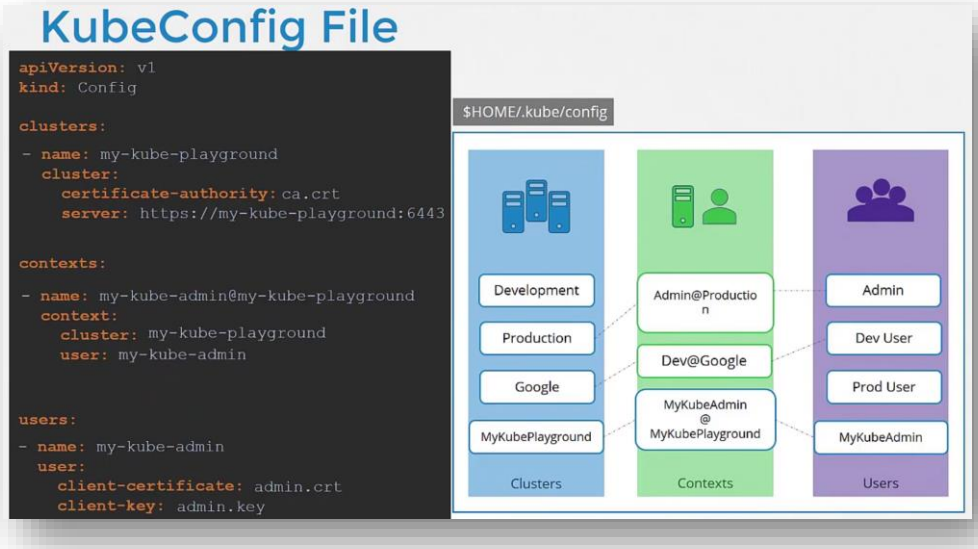
Step 2.



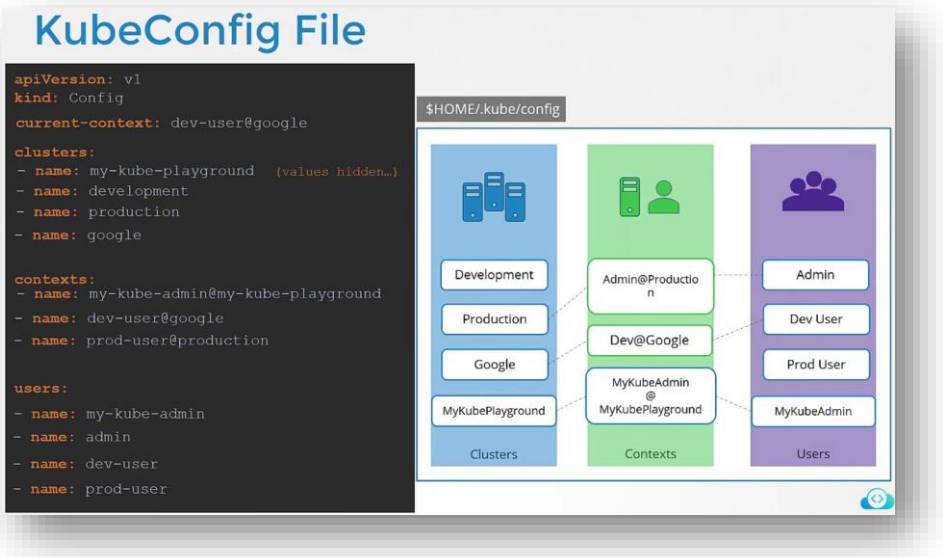
Step 3.



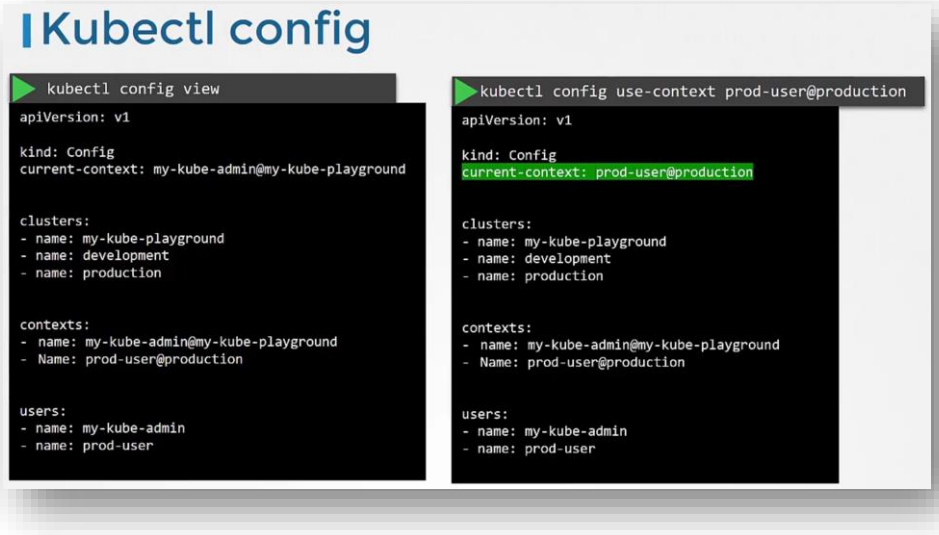
Step 4.



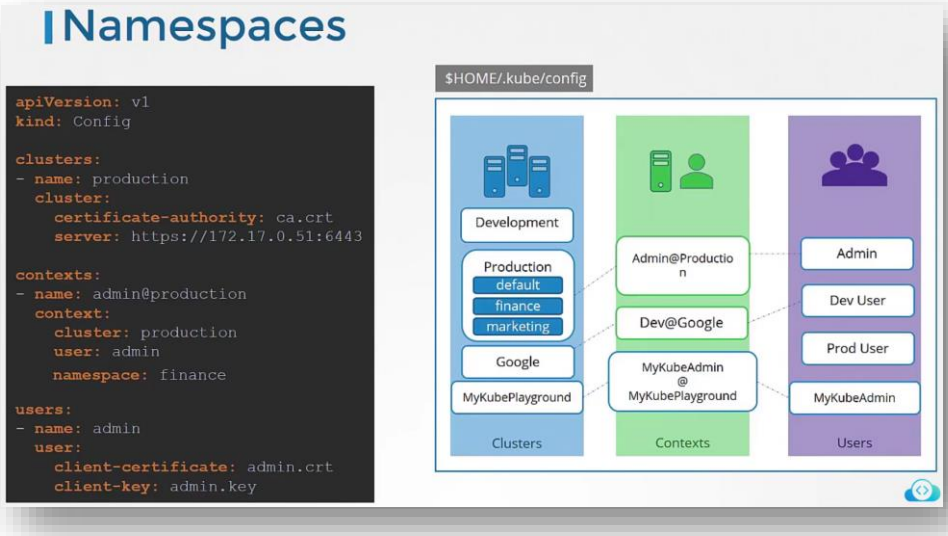
Step 5.



Step 6.

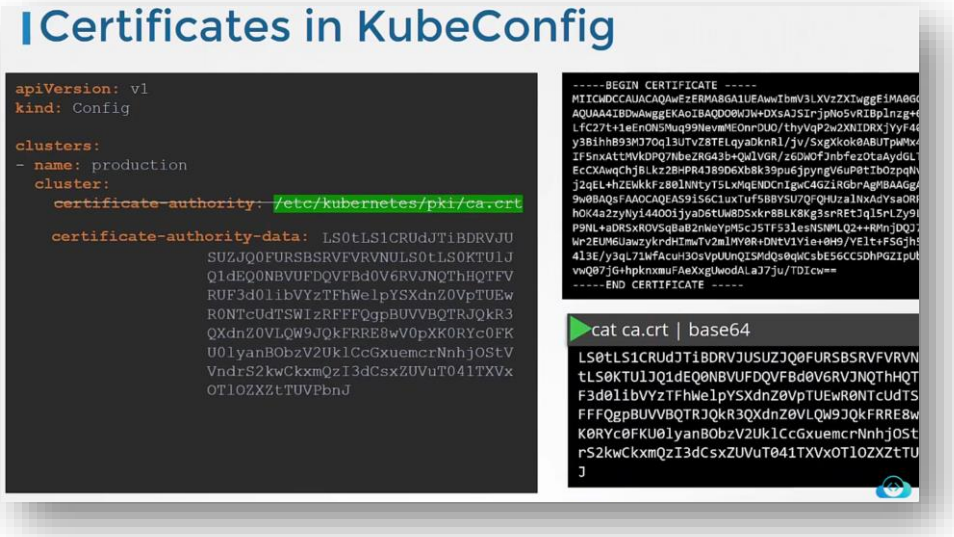


Step 7.



Step 8.

Step 9.

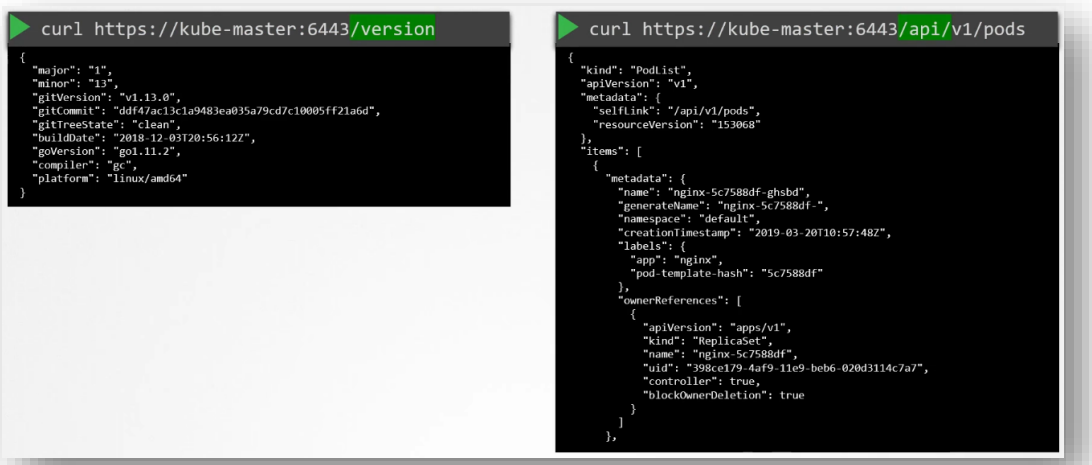


7.11.1 COMMANDLINE

- cat ca.crt | base64
- echo “LS0…bnJ” | base64 –decode
- kubectl config view
- kubectl config --kubeconfig=/root/my-kube-config use-context research
- kubectl config --kubeconfig=/root/my-kube-config current-context

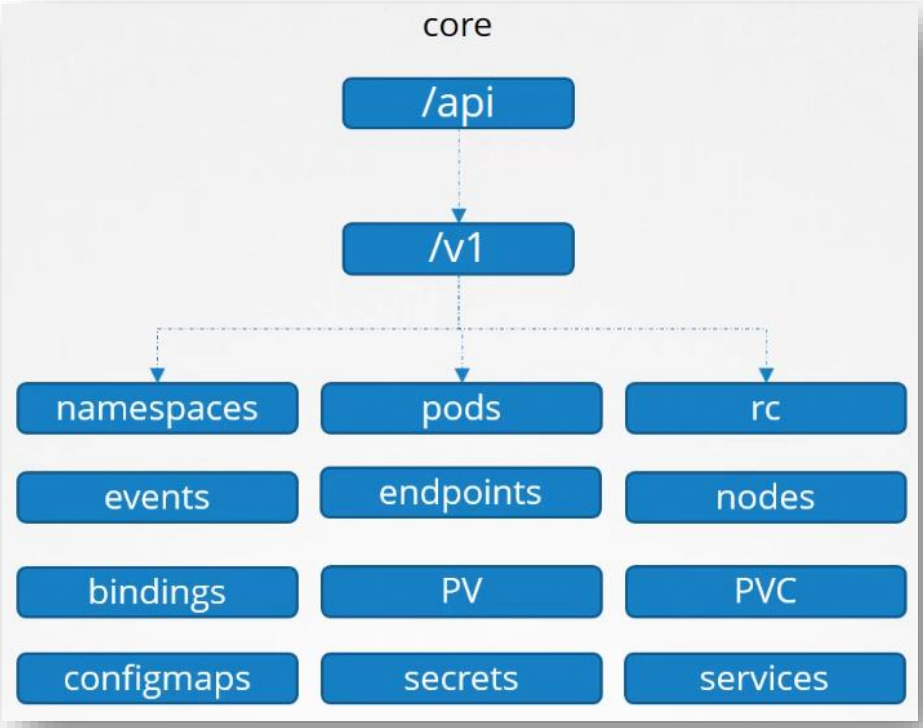
7.12 API GROUPS

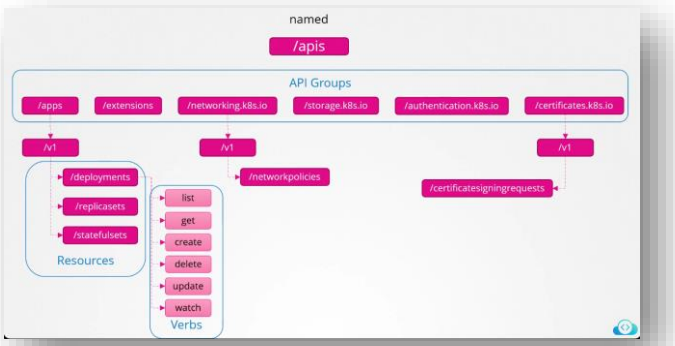
Step 10.



Step 11.

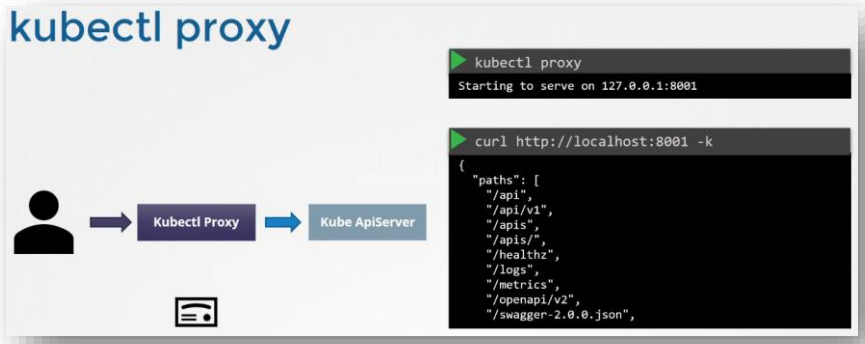
Step 12.

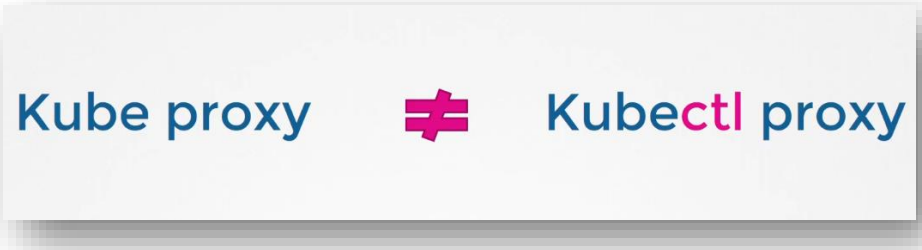




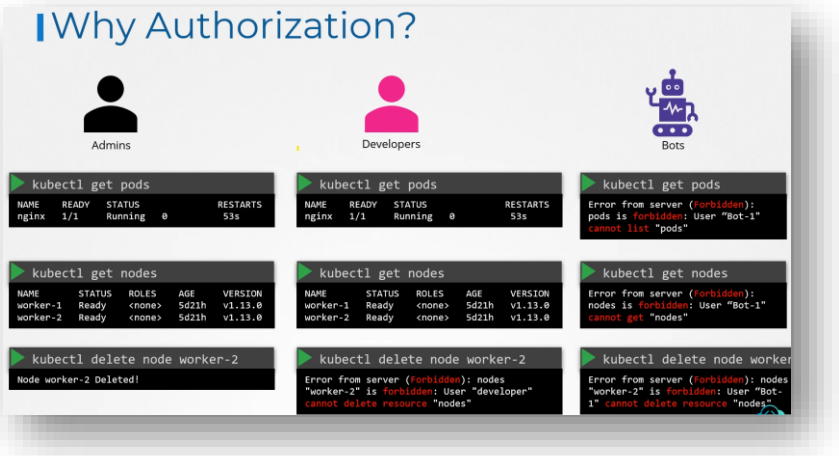

```
curl http://localhost:6443 -k
{
  "paths": [
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/healthz",
    "/logs",
    "/metrics",
    "/openapi/v2",
    "/swagger-2.0.0.json",
```

```
curl http://localhost:6443/apis -k | grep "name"
      "name": "extensions",
      "name": "apps",
      "name": "events.k8s.io",
      "name": "authentication.k8s.io",
      "name": "authorization.k8s.io",
      "name": "autoscaling",
      "name": "batch",
      "name": "certificates.k8s.io",
      "name": "networking.k8s.io",
      "name": "policy",
      "name": "rbac.authorization.k8s.io",
      "name": "storage.k8s.io",
      "name": "admissionregistration.k8s.io",
      "name": "apiextensions.k8s.io",
      "name": "scheduling.k8s.io",
```





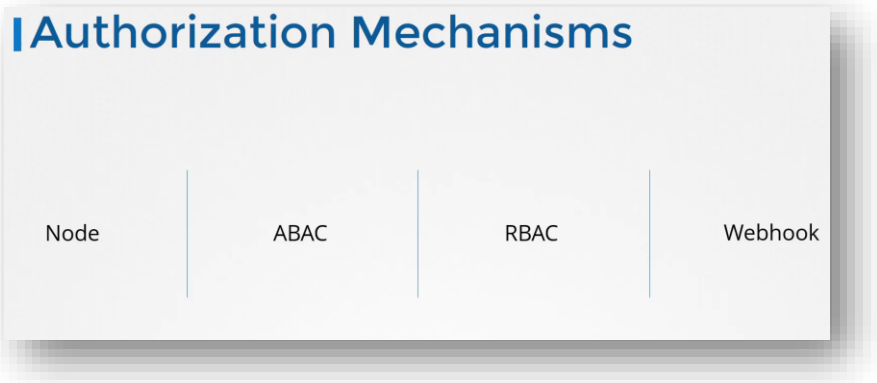
7.13 AUTHORIZATION



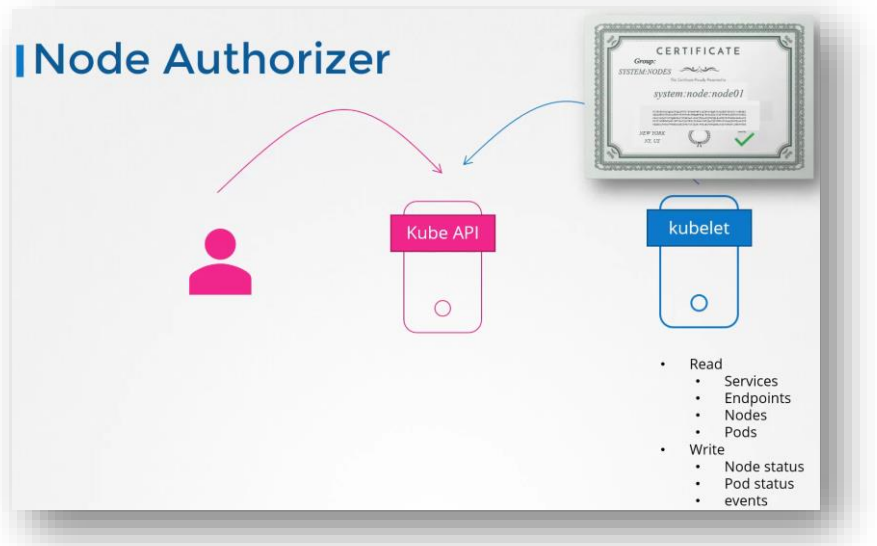
7.13.1 TYPES OF AUTHORIZATION

- Node
- ABAC (Attributes base authorization)
- RBAC (Roll base Authorization)
- Webhook

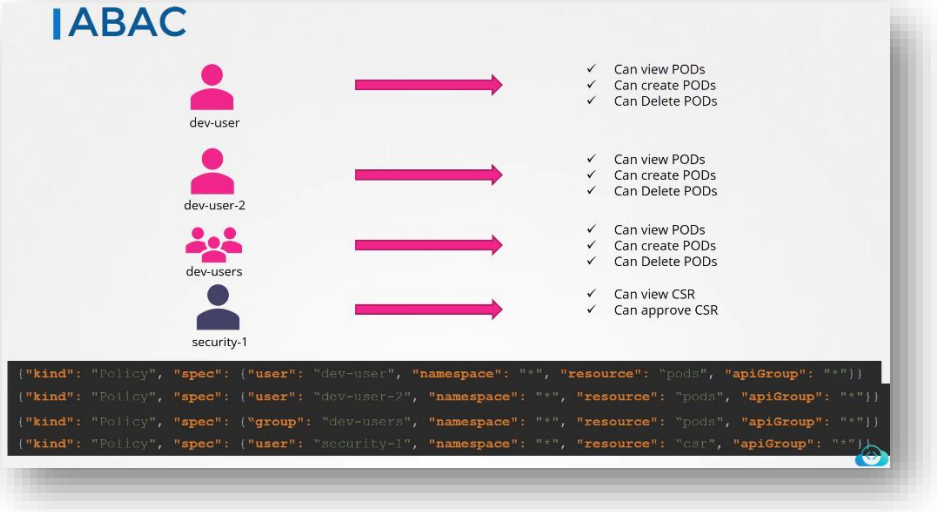
Step 13.



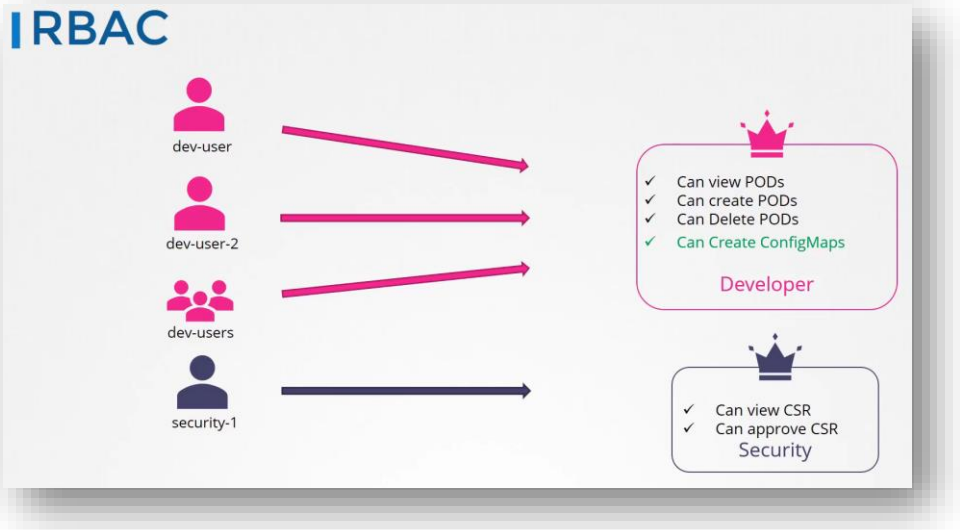
Step 14.



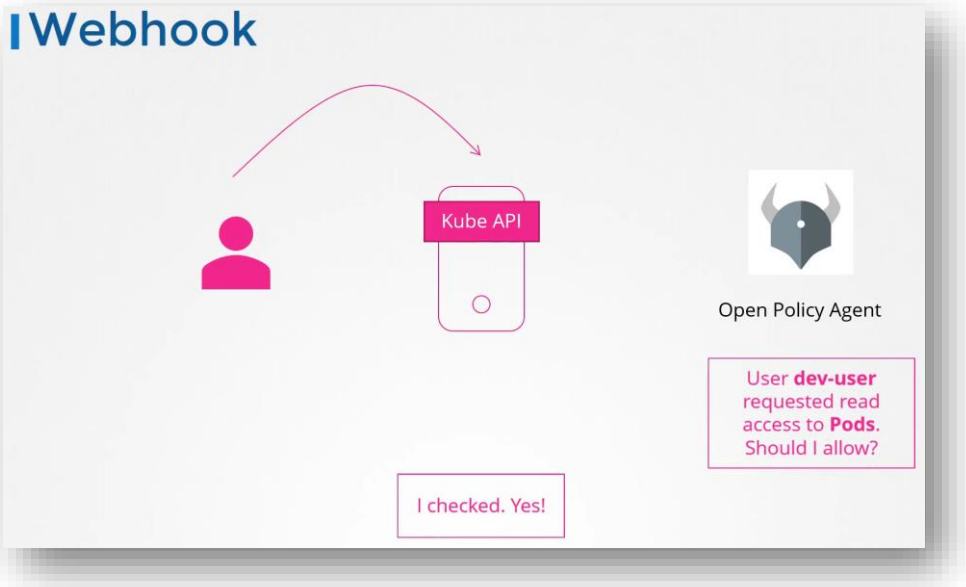
Step 15.



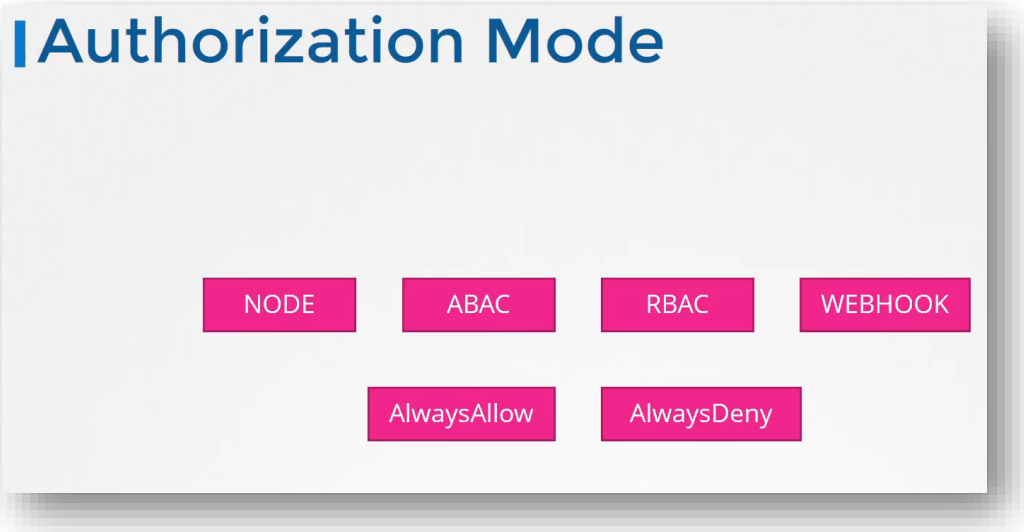
Step 16.



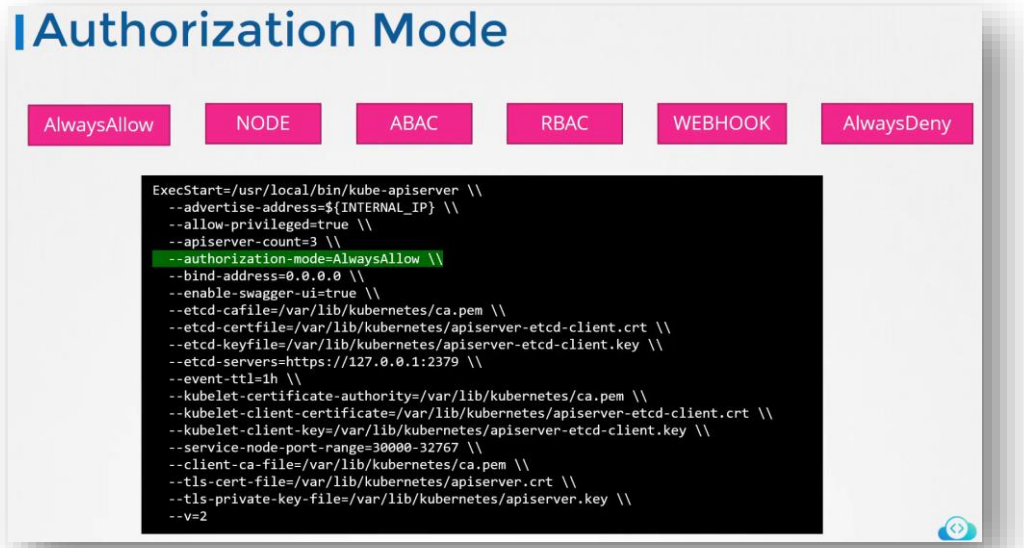
Step 17.



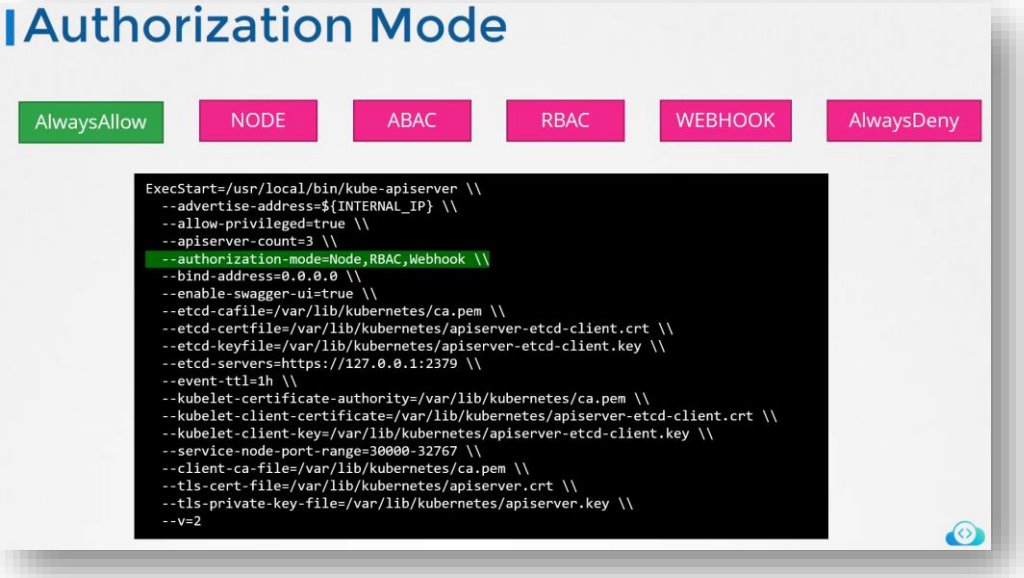
Step 18.



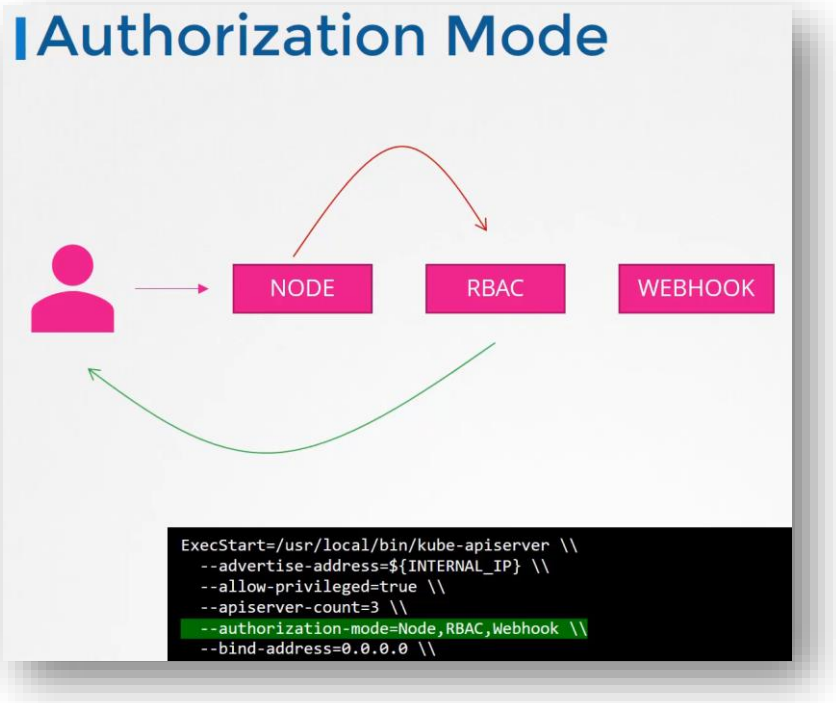
Step 19.



Step 20.

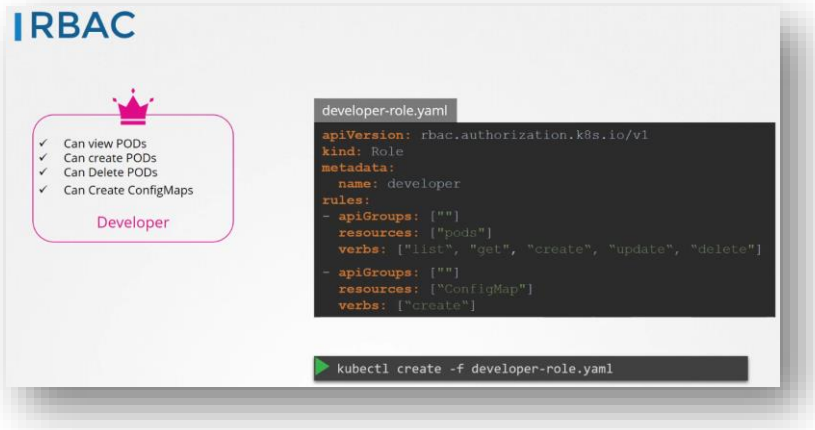


Step 21.

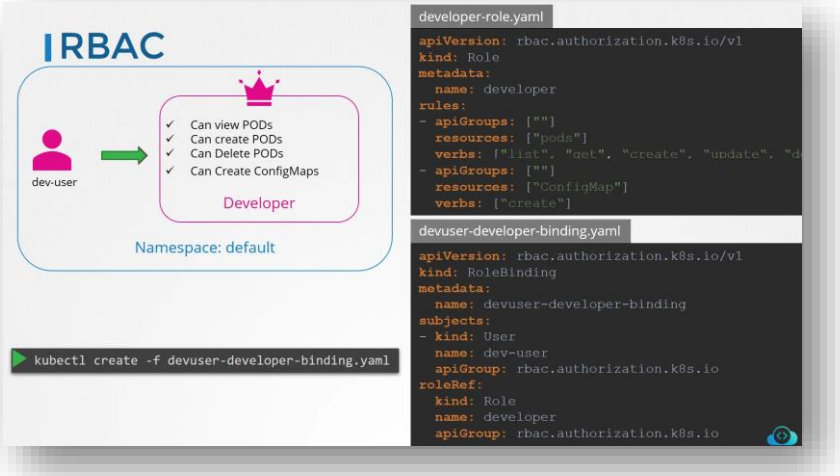


7.14 RBAC (ROLE BASED ACCESS CONTROLS)

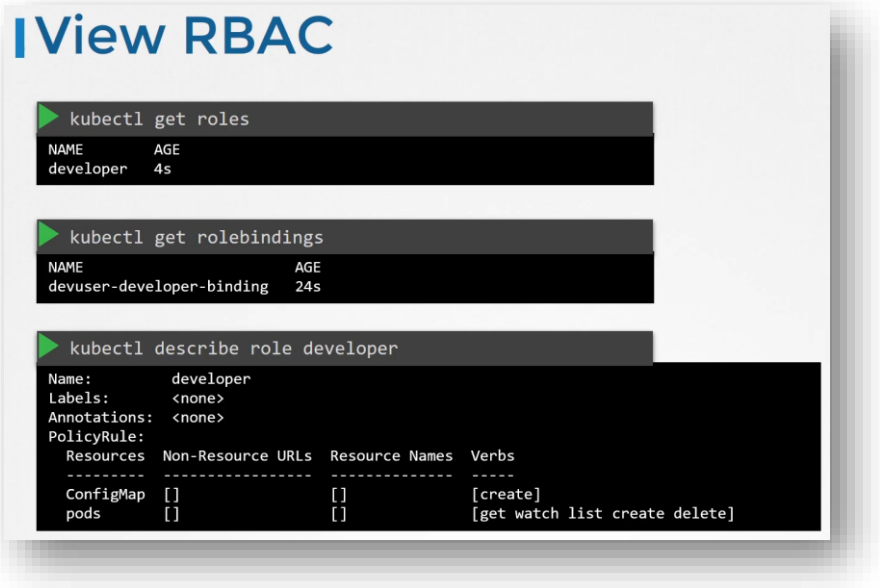
Step 22.



Step 23.

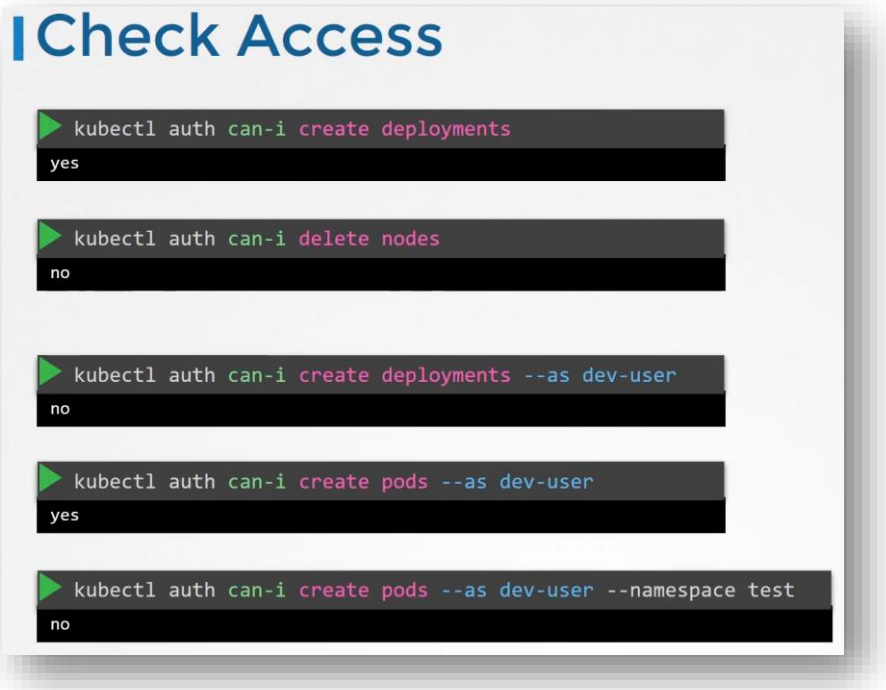


Step 24.

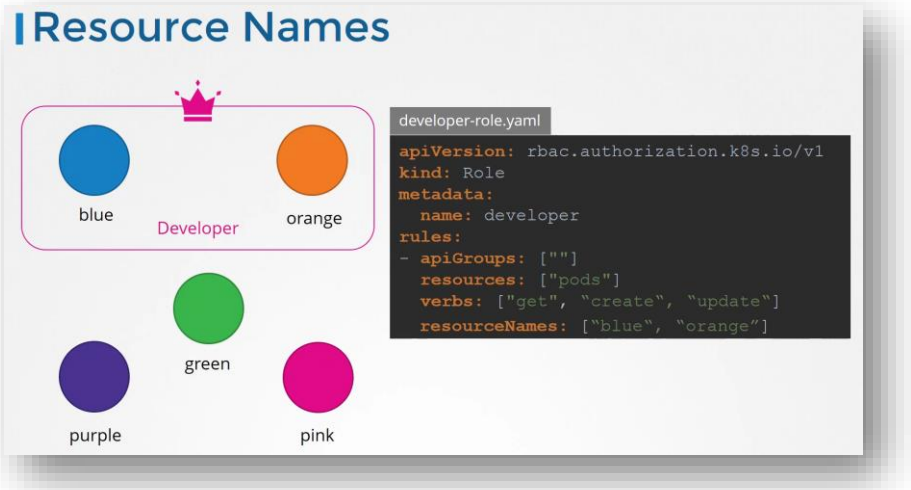


Step 25.

Step 26.



Step 27.



7.14.1 COMMANDLINE

- kubectl describe pod kube-apiserver-controlplane -n kube-system
- kubectl get roles
- kubectl get roles -n kube-system -o wide -A
- kubectl describe role kube-proxy -n kube-system

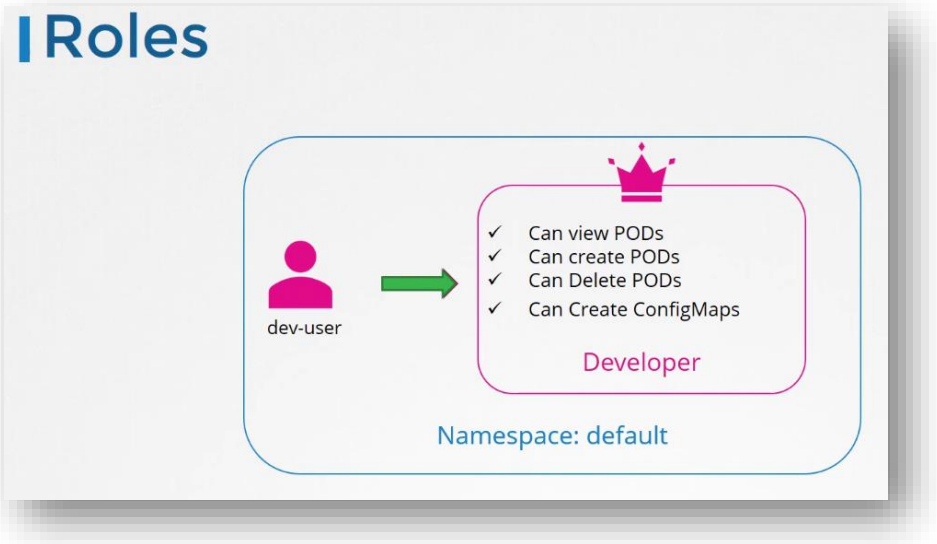
- kubectl create role developer --namespace=default --verb=list,create,delete --resource=pods

- kubectl create rolebinding dev-user-binding --namespace=default --role=developer --user=dev-user

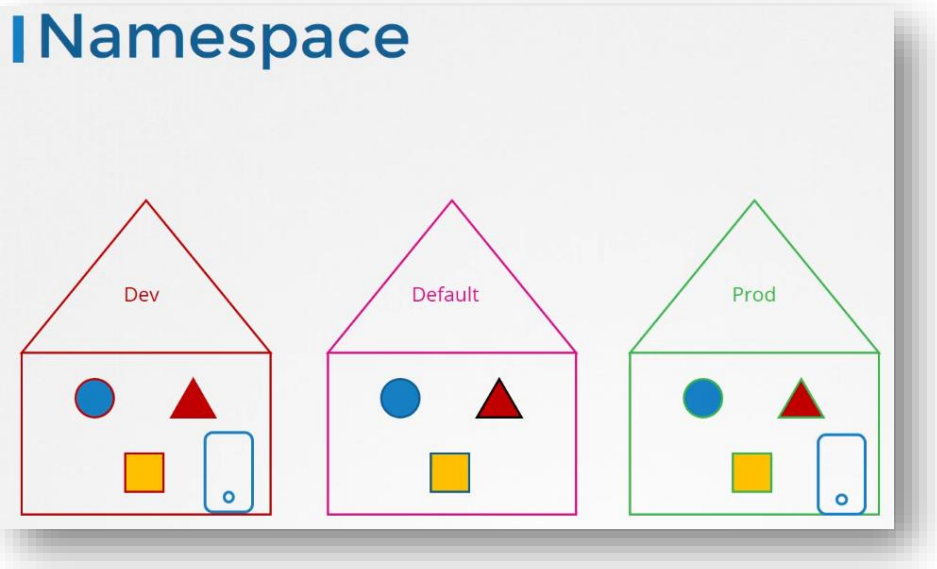
- kubectl edit role developer -n blue
- kubectl get role -o wide -A
- cat /etc/kubernetes/manifests/api-server.yaml
- kubectl get roles -A –no-headers | wl -l
- kubectl get rolebindings -n kube-system
- kubectl get pods --as dev-user
- kubectl create role --help

7.15 CLUSTER ROLES

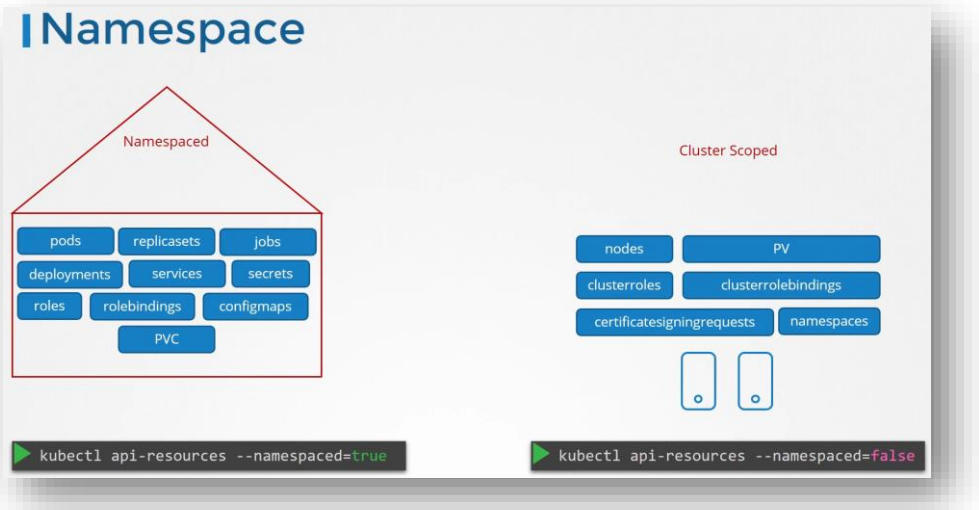
Step 28.



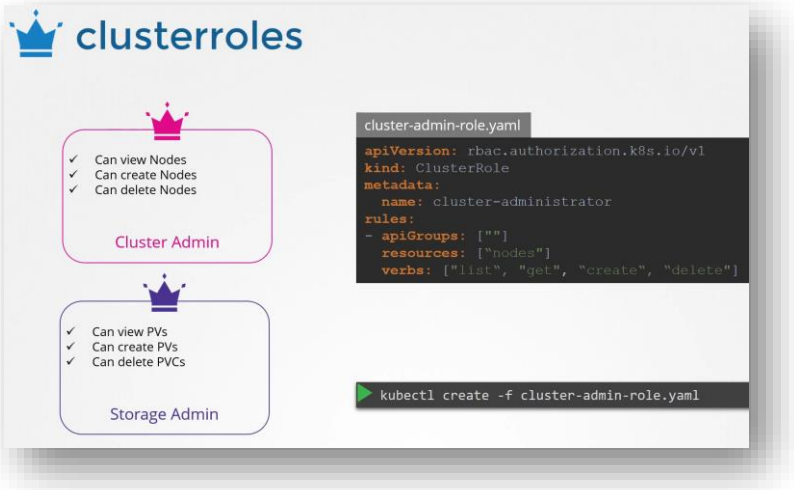
Step 29.



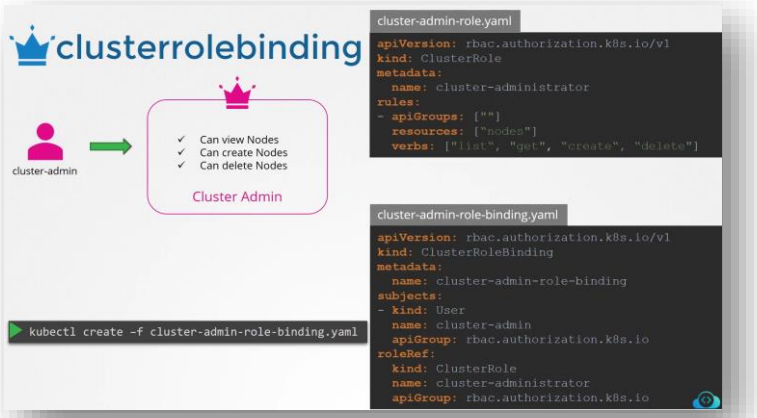
Step 30.



Step 31.



Step 32.



Step 33.

```yaml
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: node-admin
rules:
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["get", "watch", "list", "create", "delete"]

---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: michelle-binding
subjects:
- kind: User
  name: michelle
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: node-admin
  apiGroup: rbac.authorization.k8s.io
```

Step 34.

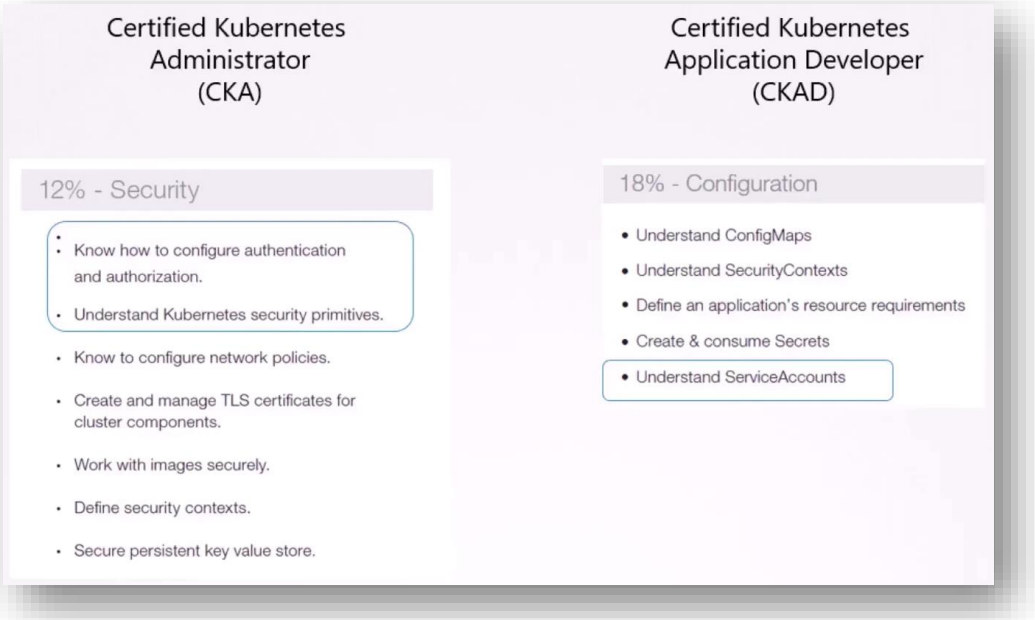
```yaml
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: storage-admin
rules:
- apiGroups: [""]
  resources: ["persistentvolumes"]
  verbs: ["get", "watch", "list", "create", "delete"]
- apiGroups: ["storage.k8s.io"]
  resources: ["storageclasses"]
  verbs: ["get", "watch", "list", "create", "delete"]

---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: michelle-storage-admin
subjects:
- kind: User
  name: michelle
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: storage-admin
  apiGroup: rbac.authorization.k8s.io
```

7.15.1 COMMANDLINE

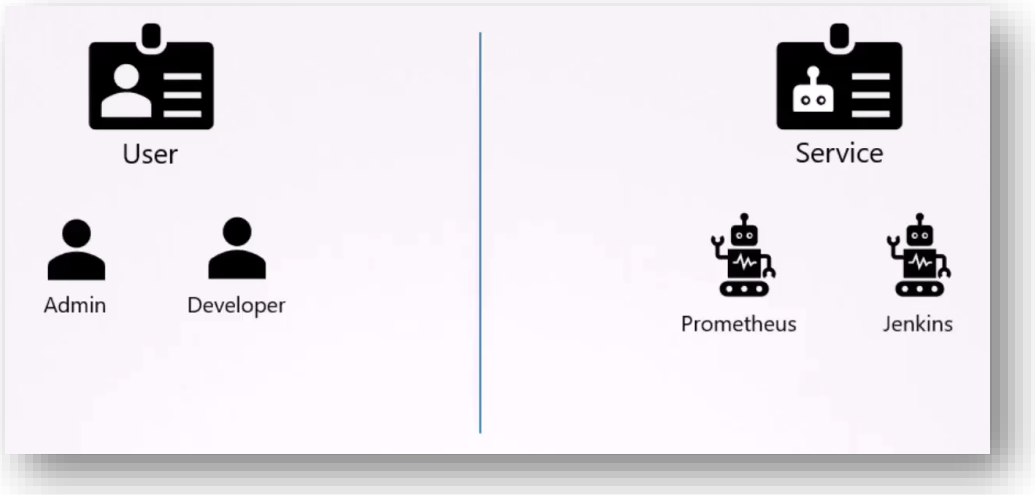
- kubectl get clusterroles --no-headers | wc -l
  (-c=Count bytes, -m=Count characters, -l=Count newlines, -w= Count words -L=Print longest line length)
- kubectl get clusterrolebindings --no-headers | wc -l
- kubectl api-resources --namespaced=false
- kubectl describe clusterrolebinding cluster-admin
- kubectl describe clusterrole cluster-admin
- kubectl clusterrolebindings | grep cluster-admin
- kubectl get nodes –as michelle
- kubectl create clusterrole --help
- kubectl create clusterrole michele-role –verb=get,list,watch –resource=nodes
- kubectl create clusterrolebinding michel-role-binding –clusterrole=michelle-role –user=michelle
- kubectl describe clusterrole michelle-role
- kubectl describe clusterrolebinding michelle-role-binding
- kubectl auth can-i list nodes --as michelle
- kubectl api=resources
- kubectl create clusterrole storage-admin –resource=persistentvolumes,storageclasses –verb=list,create,get,watch
- kubectl create clusterrolebinding michelle-storage-admin –user=michelle –clusterrole=storage-admin
- kubectl describe clusterrolebinding michelle-storage-admin

7.16 SERVICE ACCOUNTS

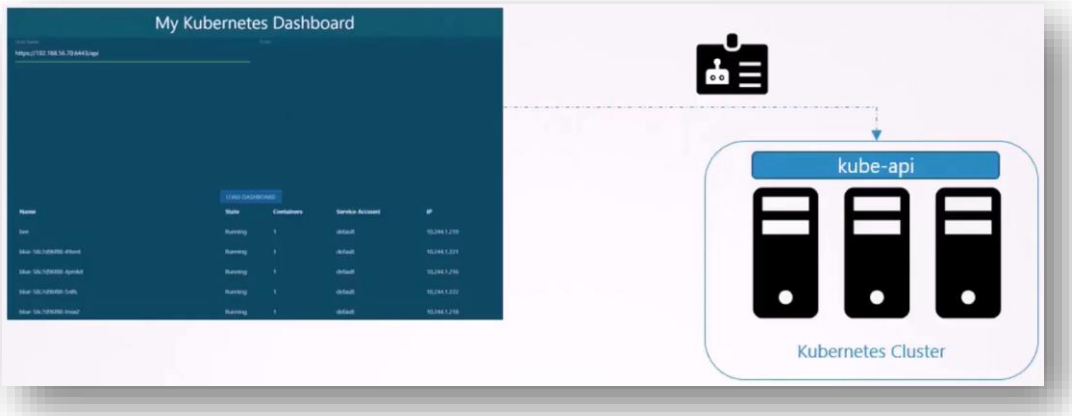
Step 35.



Step 36.



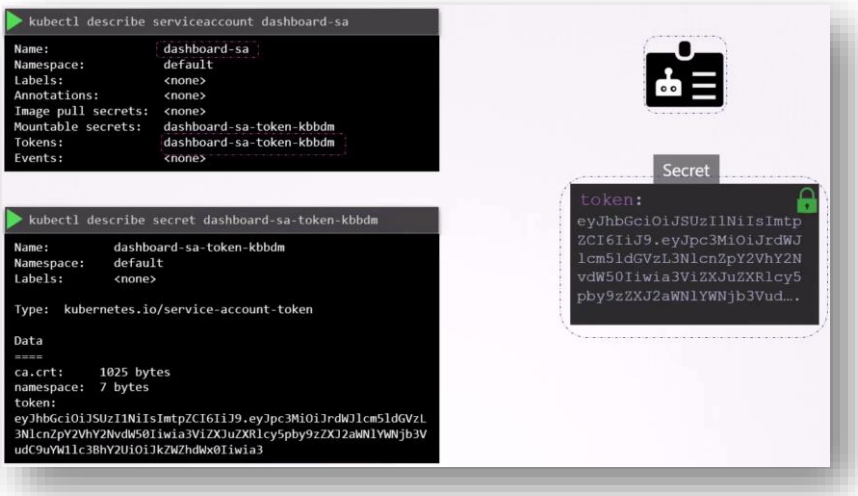
Step 37.



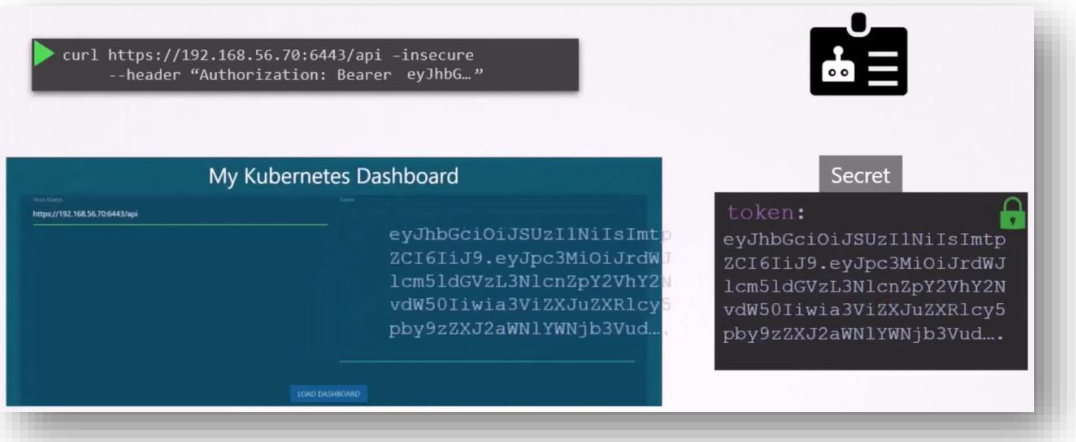
Step 38.

```
kubectl create serviceaccount dashboard-sa
serviceaccount “dashboard-sa” created

kubectl get serviceaccount
NAME                  SECRETS   AGE
default               1         218d
dashboard-sa          1         4d

kubectl describe serviceaccount dashboard-sa
Name:                 dashboard-sa
Namespace:            default
Labels:               <none>
Annotations:          <none>
Image pull secrets:   <none>
Mountable secrets:    dashboard-sa-token-kbbdm
Tokens:               dashboard-sa-token-kbbdm
Events:               <none>
```

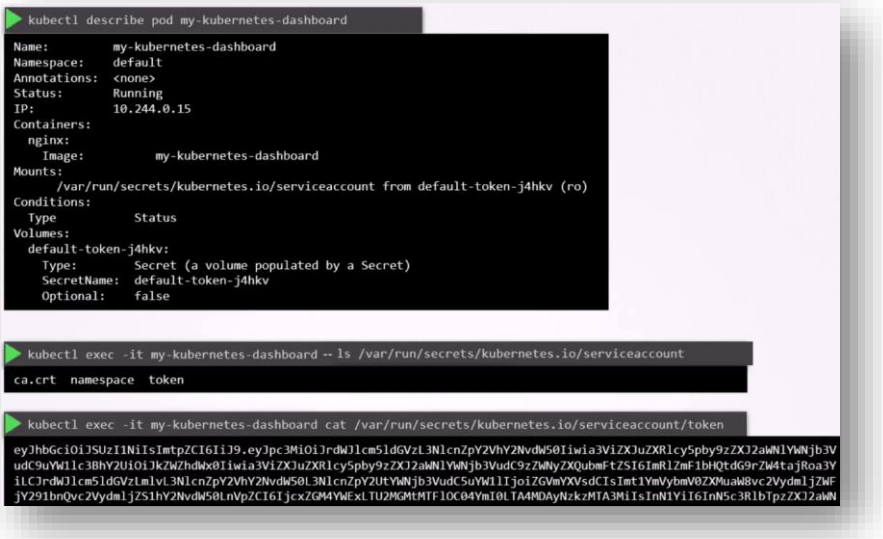
Step 39.



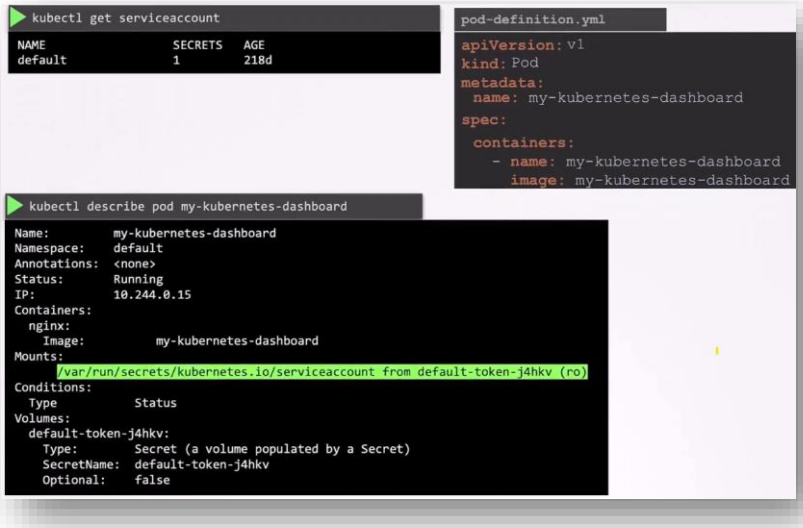
Step 40.



Step 41.



Step 42.

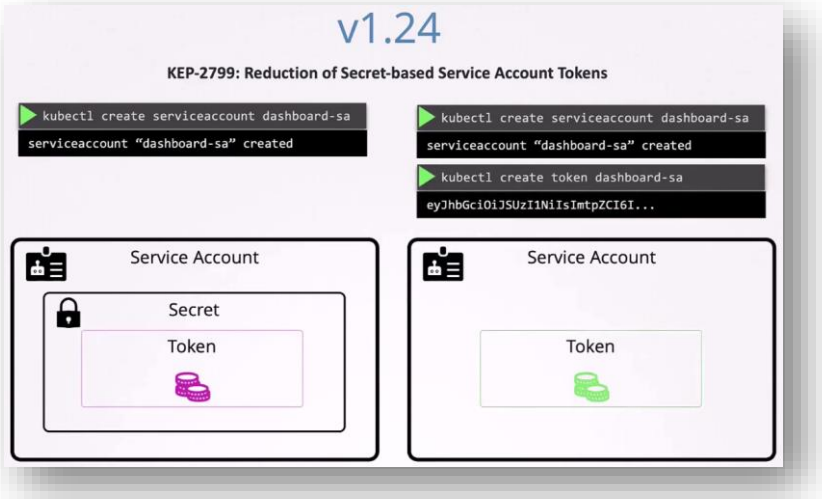


Step 43.

Step 44.

```
kubectl get pod my-kubernetes-dashboard -o yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: default
spec:
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-6mtg8
      readOnly: true
  volumes:
  - name: kube-api-access-6mtg8
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          expirationSeconds: 3607
          path: token
      - configMap:
          items:
          - key: ca.crt
            path: ca.crt
          name: kube-root-ca.crt
      - downwardAPI:
          items:
          - fieldRef:
              apiVersion: v1
```

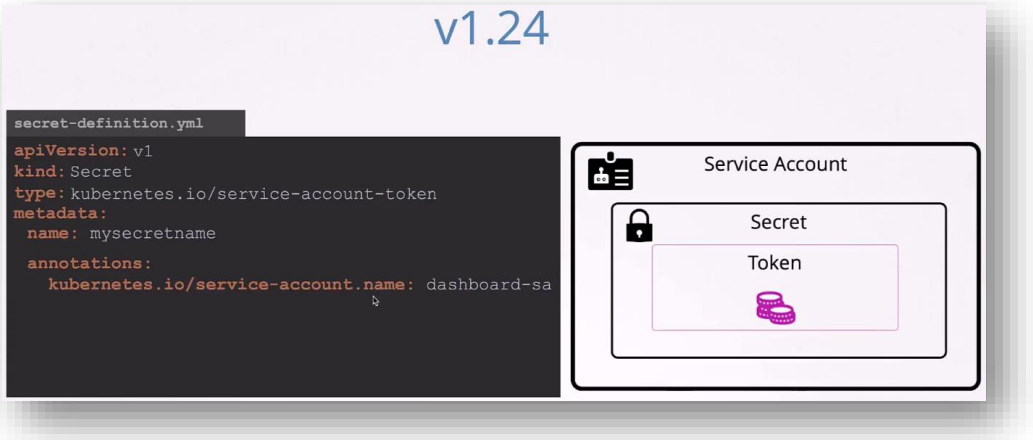
Step 45.



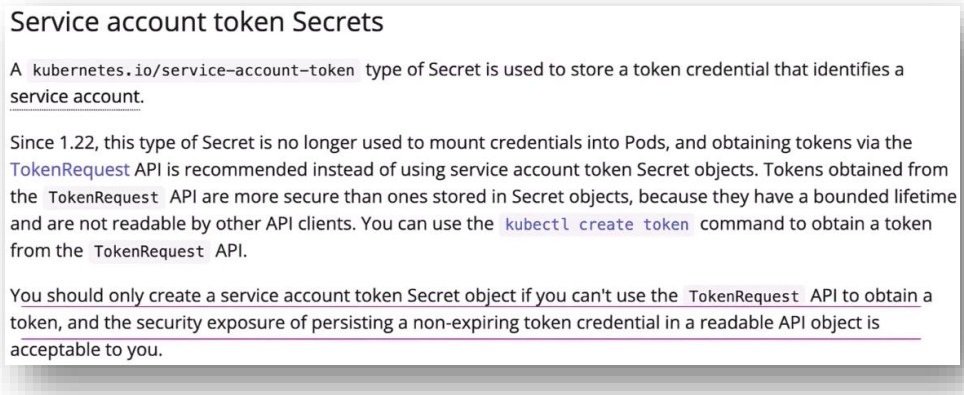
Step 46.

```
jq -R 'split(".") | select(length > 0) | .[0],.[1] | @base64d | fromjson' <<< eyJhbGciOiJSUz...
{
  "alg": "RS256",
  "kid": "vOp_YyzJsnmxNV9uQ8zzs0LIHHa0n3Dy80OvlEjXnSs"
}
{
  "aud": [
    "https://kubernetes.default.svc.cluster.local"
  ],
  "exp": 1664037763,
  "iat": 1664034163,
  "iss": "https://kubernetes.default.svc.cluster.local",
  "kubernetes.io": {
    "namespace": "default",
    "serviceaccount": {
      "name": "dashboard-sa",
      "uid": "7d0fdfe8-fbf3-4ff9-b362-d004297a73f6"
    }
  },
  "nbf": 1664034163,
  "sub": "system:serviceaccount:default:dashboard-sa"
}
```

Step 47.



Step 48.



Service account token Secrets

A `kubernetes.io/service-account-token` type of Secret is used to store a token credential that identifies a service account.

Since 1.22, this type of Secret is no longer used to mount credentials into Pods, and obtaining tokens via the TokenRequest API is recommended instead of using service account token Secret objects. Tokens obtained from the `TokenRequest` API are more secure than ones stored in Secret objects, because they have a bounded lifetime and are not readable by other API clients. You can use the `kubectl create token` command to obtain a token from the `TokenRequest` API.

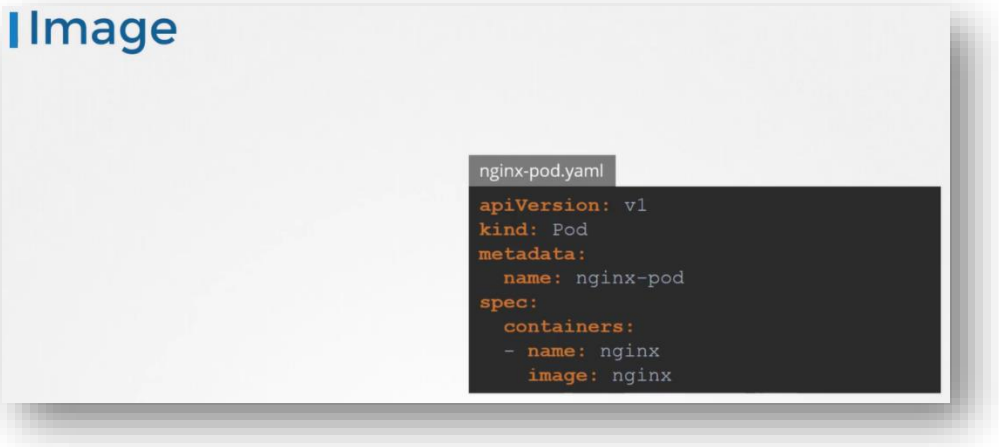
You should only create a service account token Secret object if you can't use the `TokenRequest` API to obtain a token, and the security exposure of persisting a non-expiring token credential in a readable API object is acceptable to you.

7.16.1 COMMANDLINE

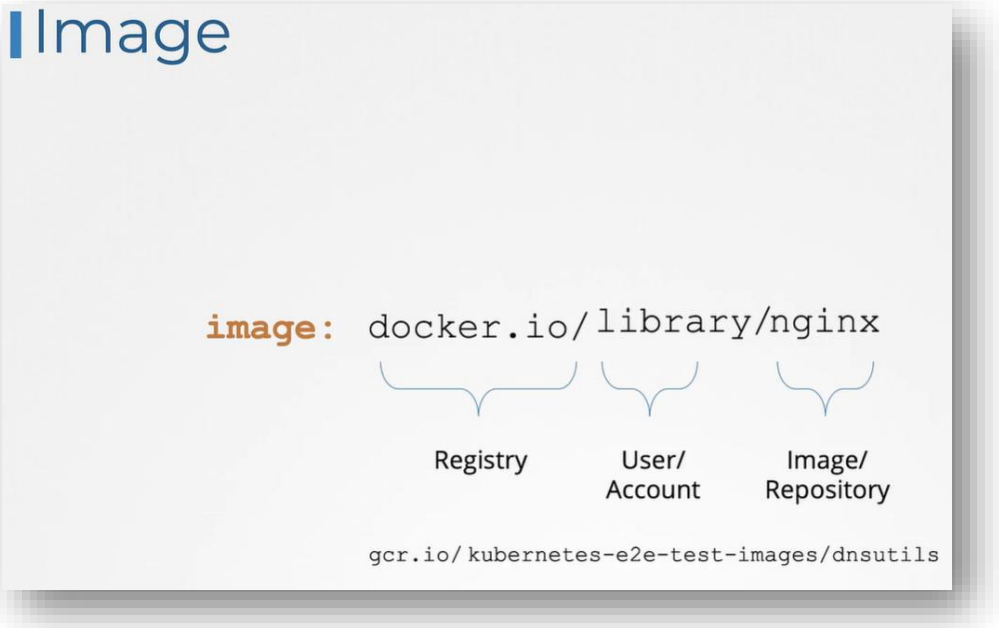
- kubectl get serviceaccount
- kubectl describe serviceaccounts default
- /var/run/secrets/kubernetes.io/serviceaccount
- Kubectl create serviceaccount dashboard-sa
- kubectl set serviceaccount deploy/web-dashboard dashboard-sa

7.17 IMAGE SECURITY

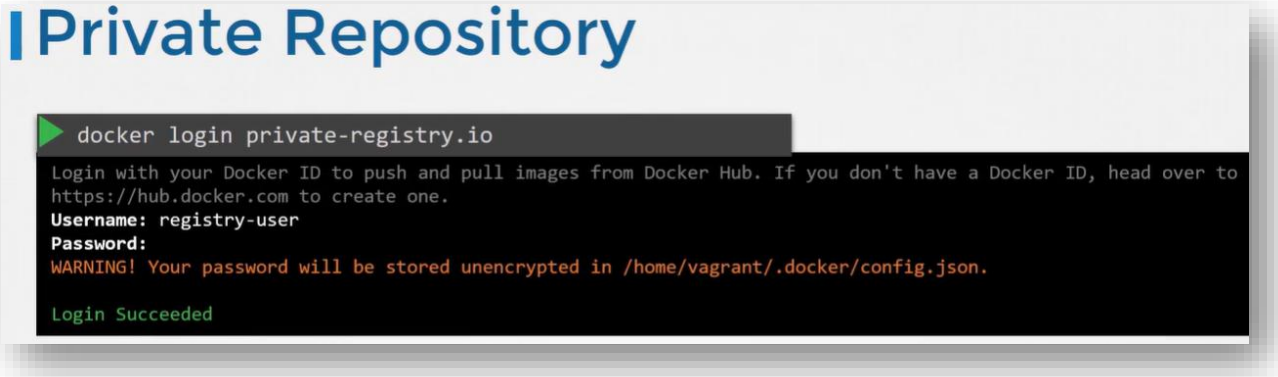
Step 1.



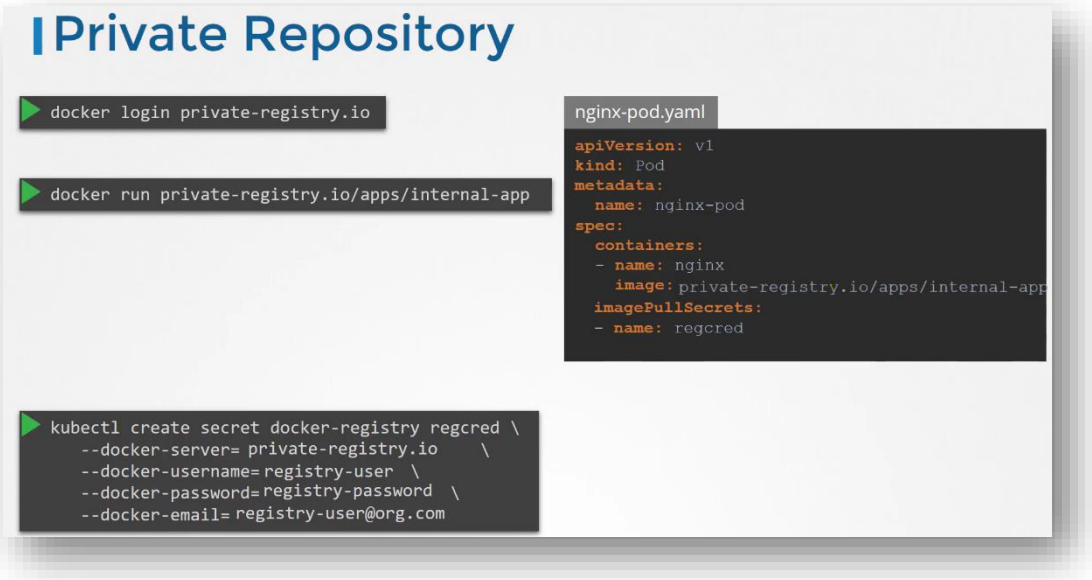
Step 2.



Step 3.



Step 4.

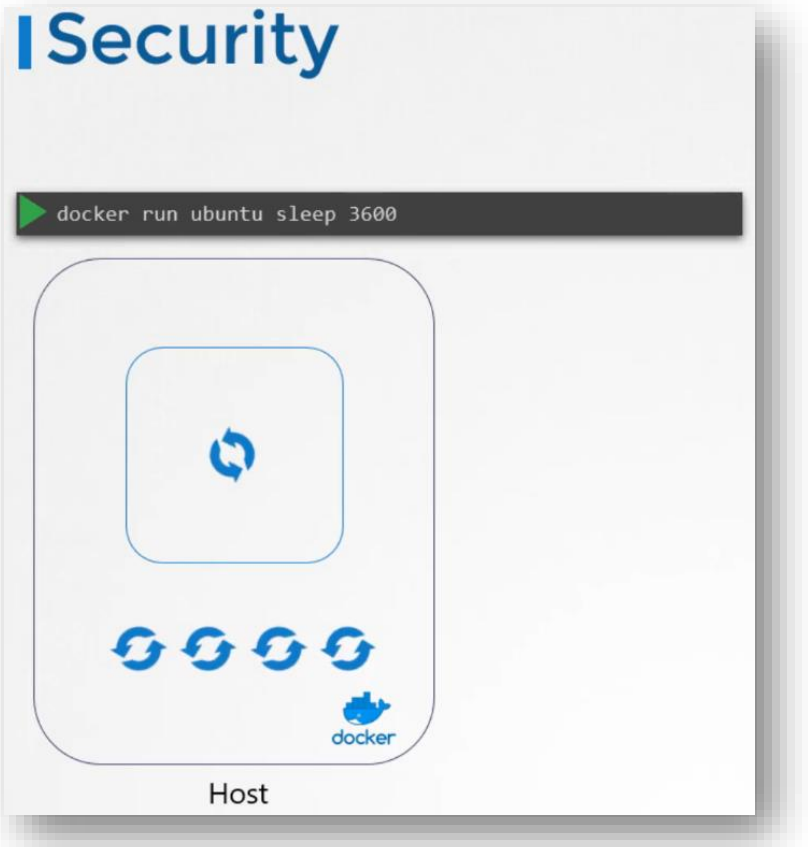


7.17.1 COMMANDLINE

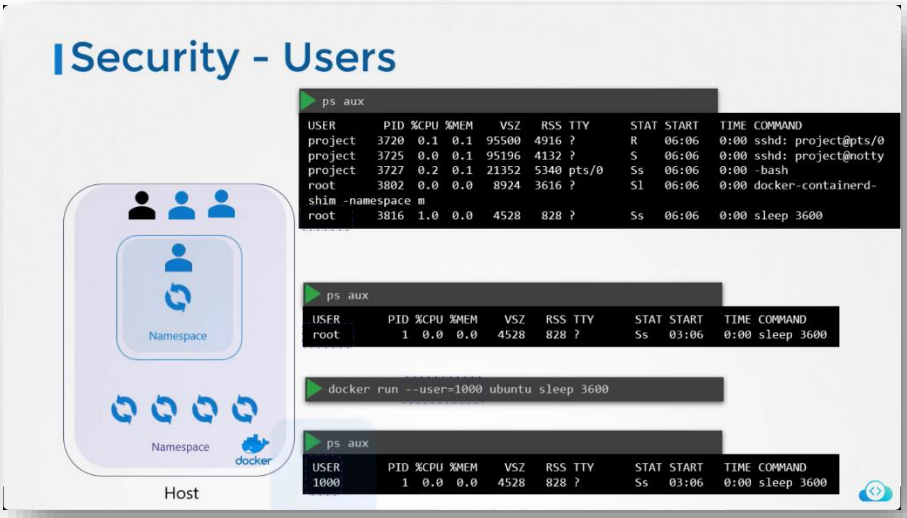
- kubectl create secret
- alias k=kubectl

7.18 PRE-REQUISITE – SECURITY IN DOCKER

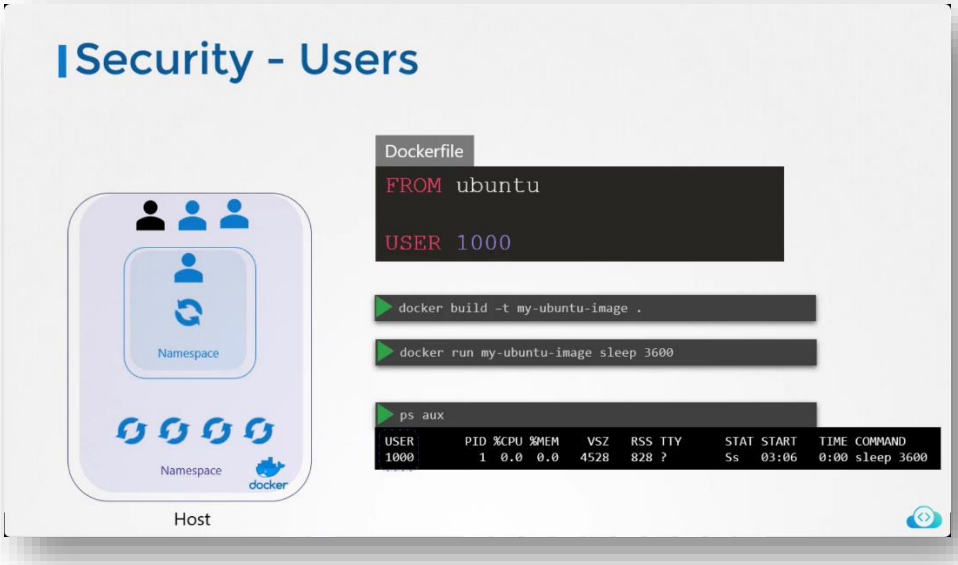
Step 1.



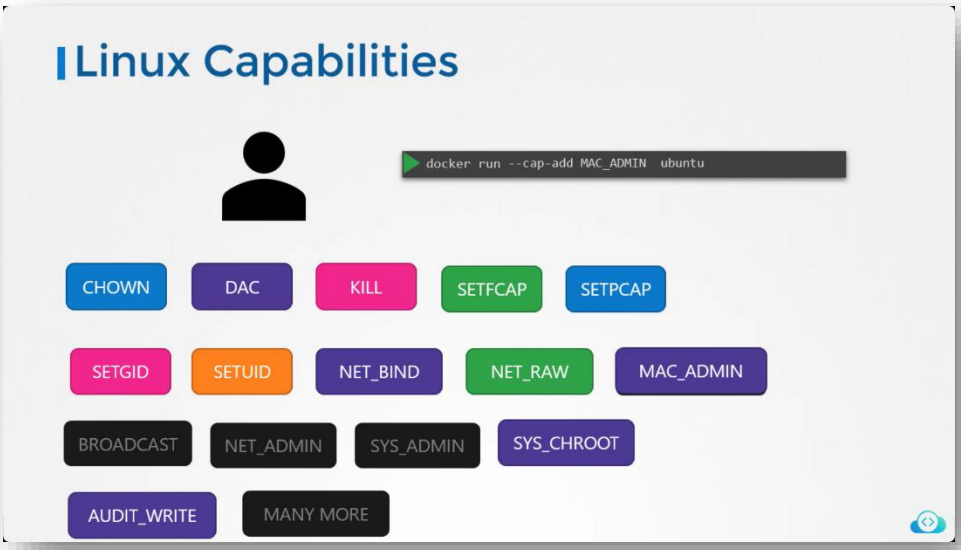
Step 2.



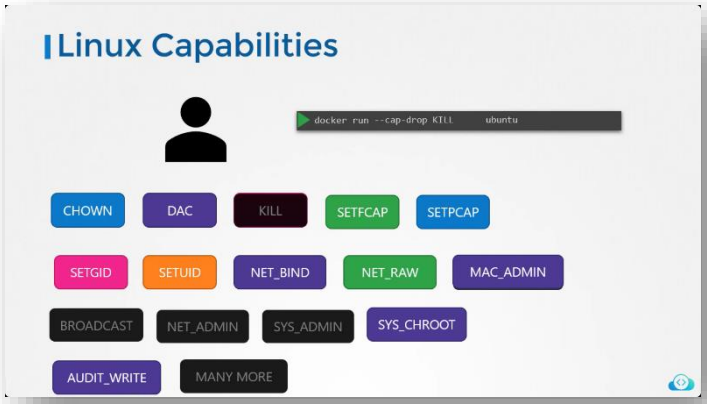
Step 3.



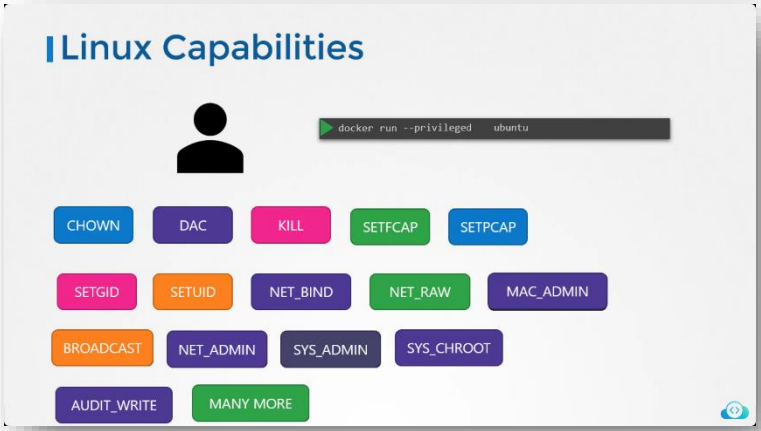
Step 4.



Step 5.

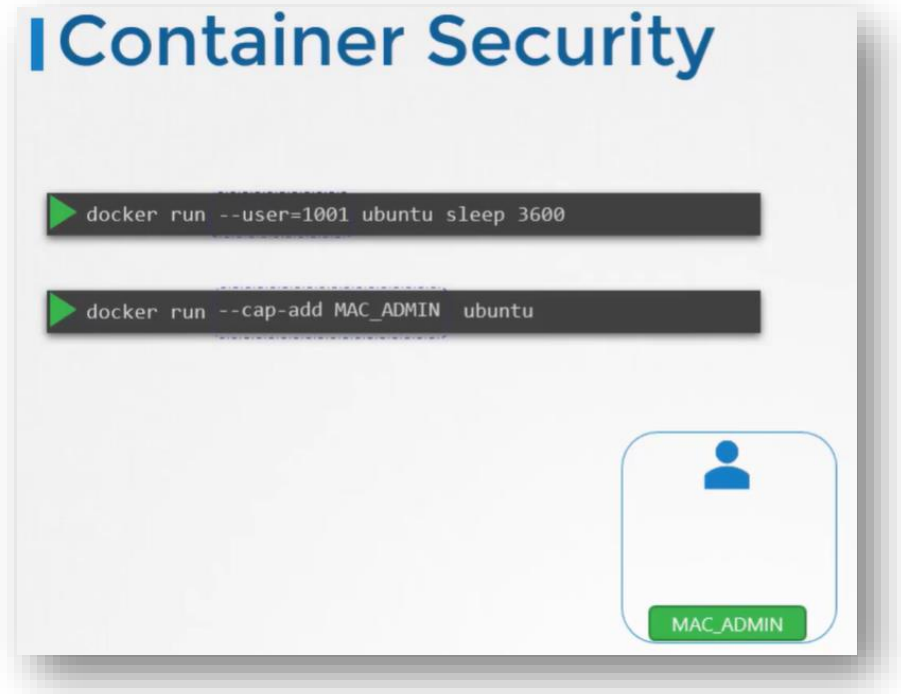


Step 6.

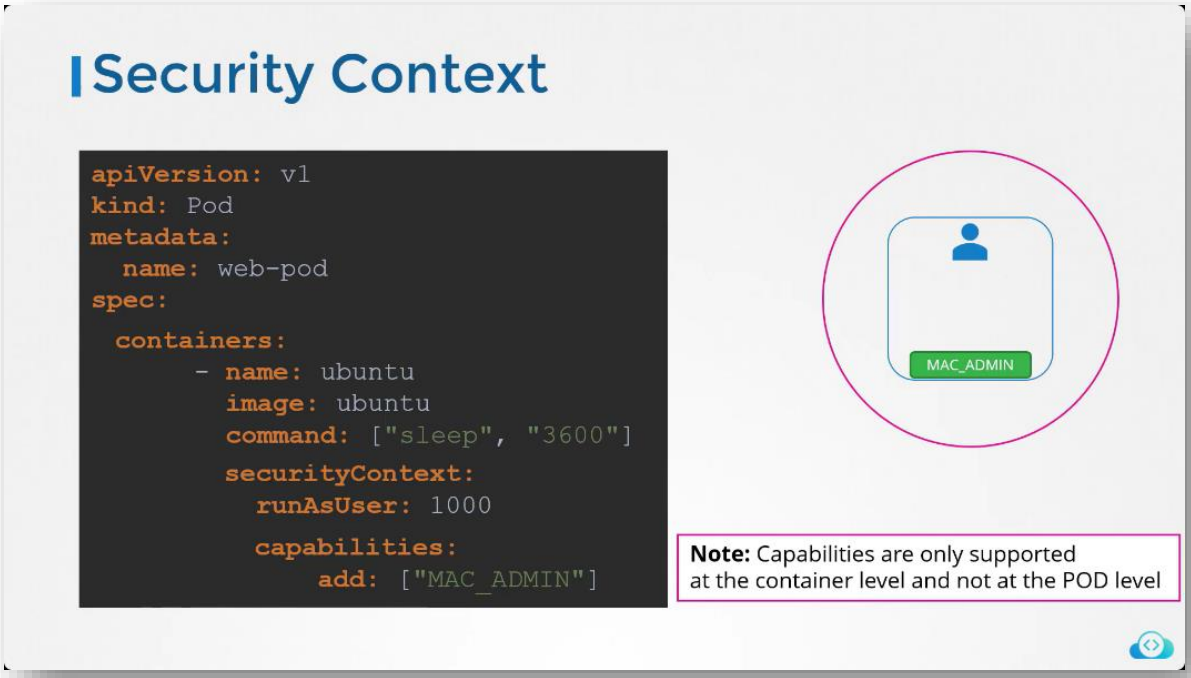


7.19 SECURITY CONTEXTS

Step 7.



Step 8.



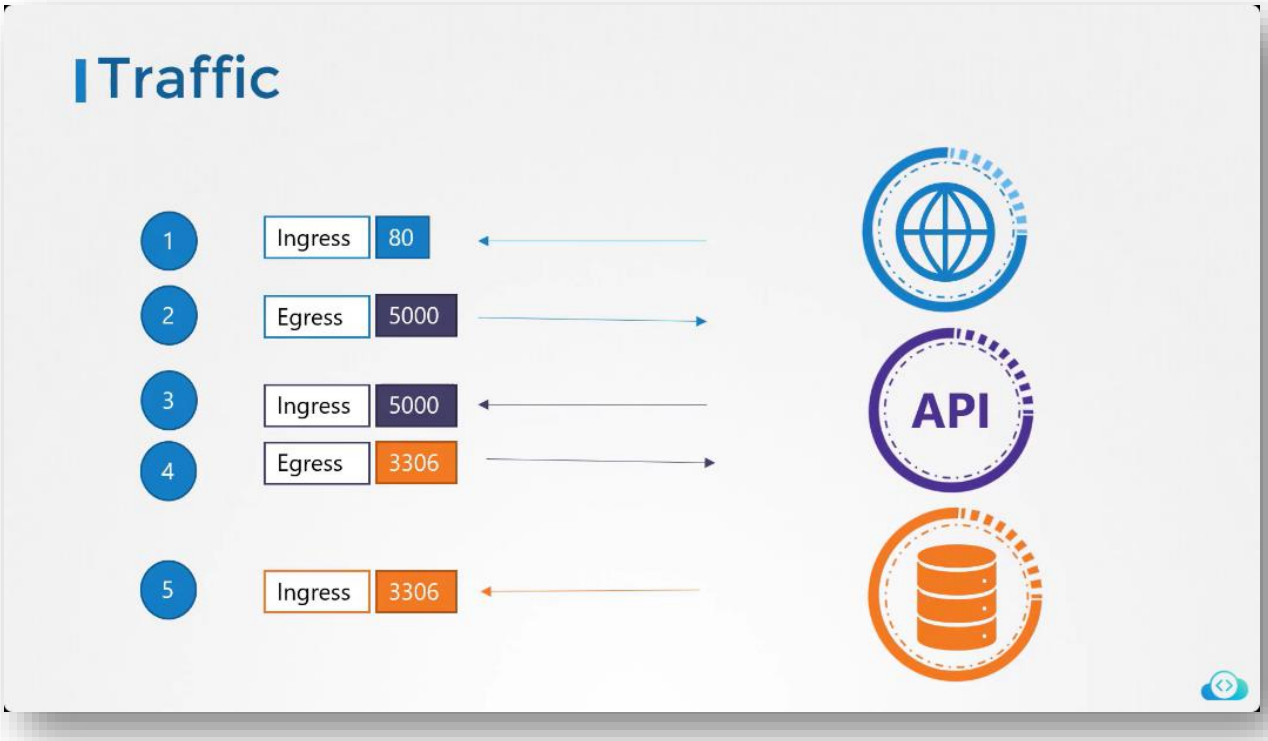
7.19.1 COMMANDLINE

- kubectl exec ubuntu-sleeper – whoami
-

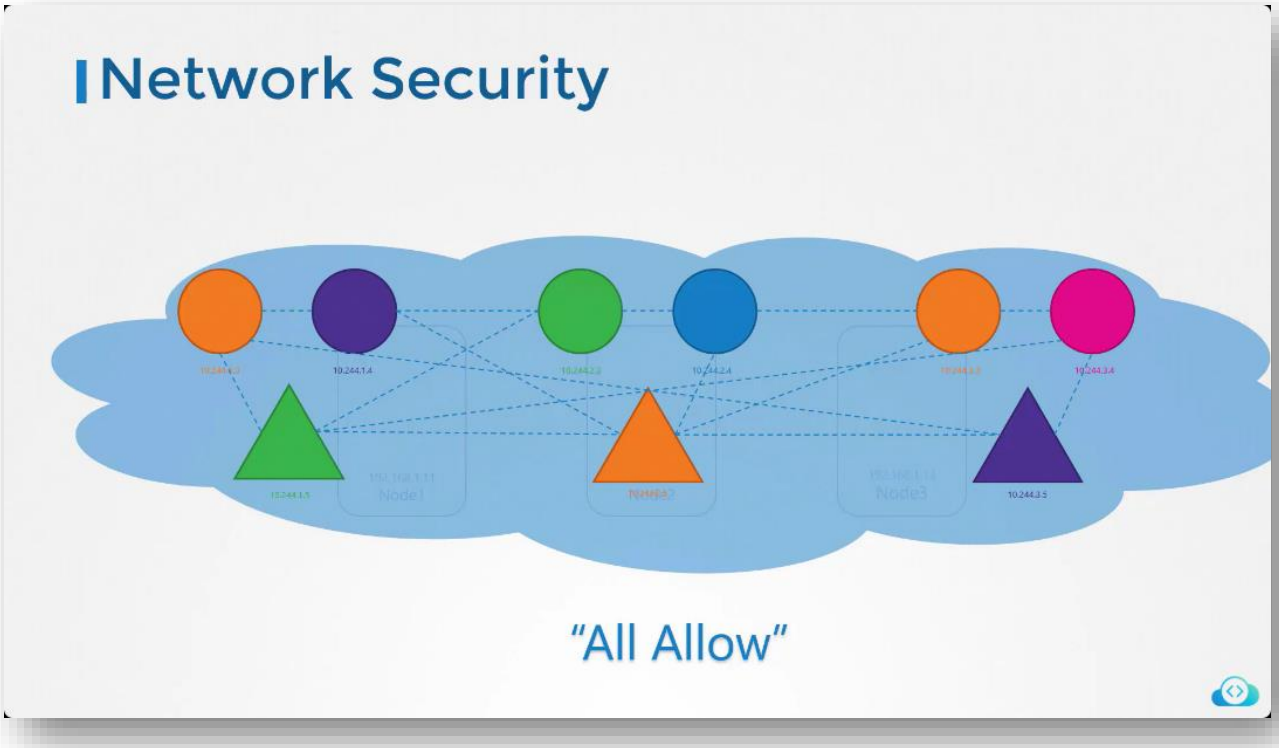
7.20 NETWORK POLICIES

Step 1.

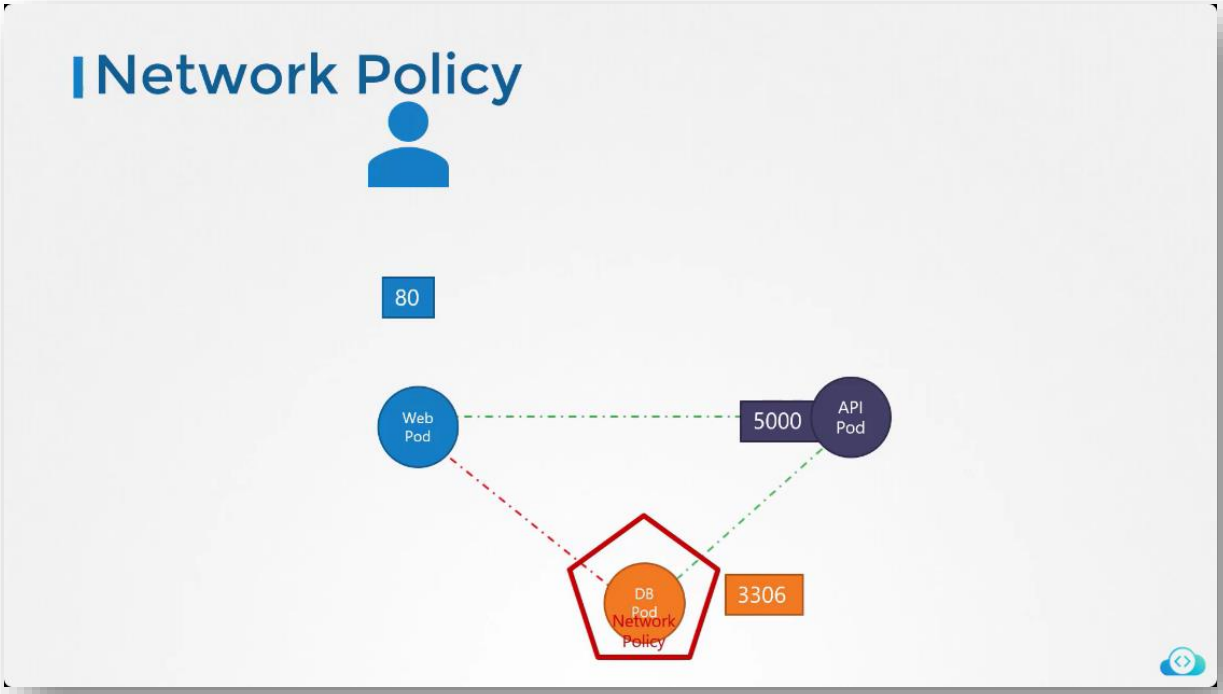
Step 2.


Traffic
1
Ingress
80
2
Egress
5000
3
Ingress
5000
4
Egress
3306
5
Ingress
3306
API

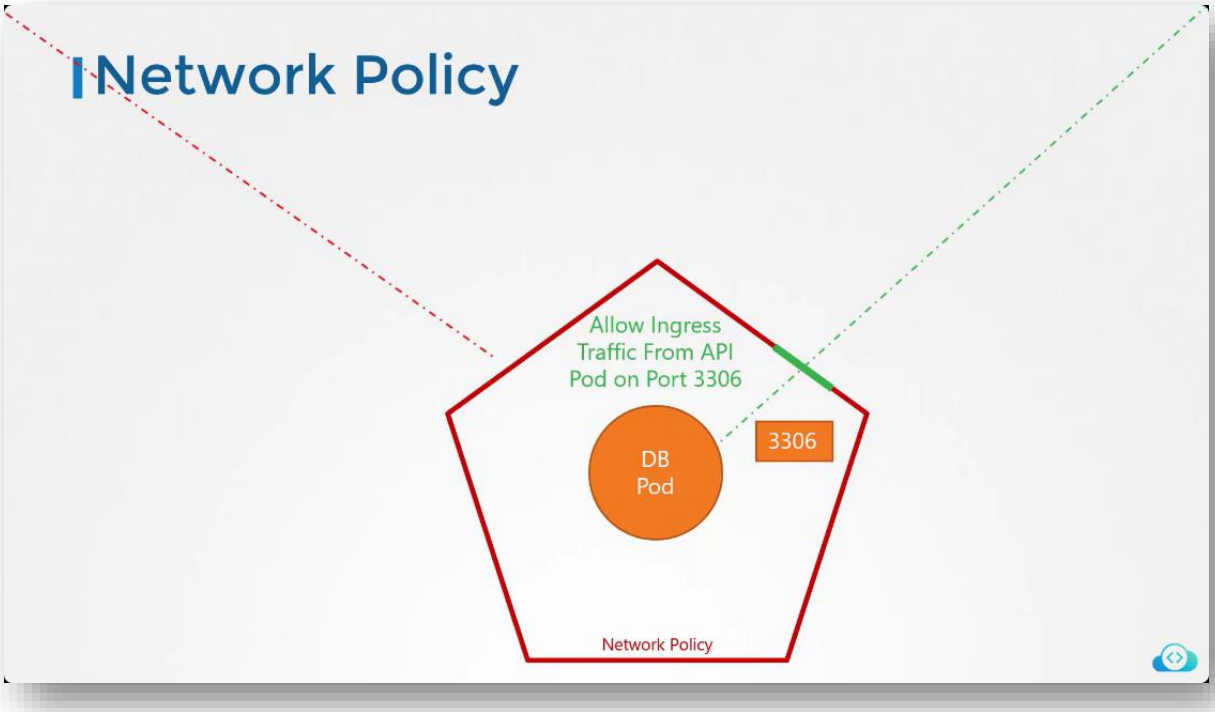
Step 3.


Network Security
"All Allow"

Step 4.


Network Policy
80
Web Pod
5000
API Pod
DB Pod
Network Policy
3306

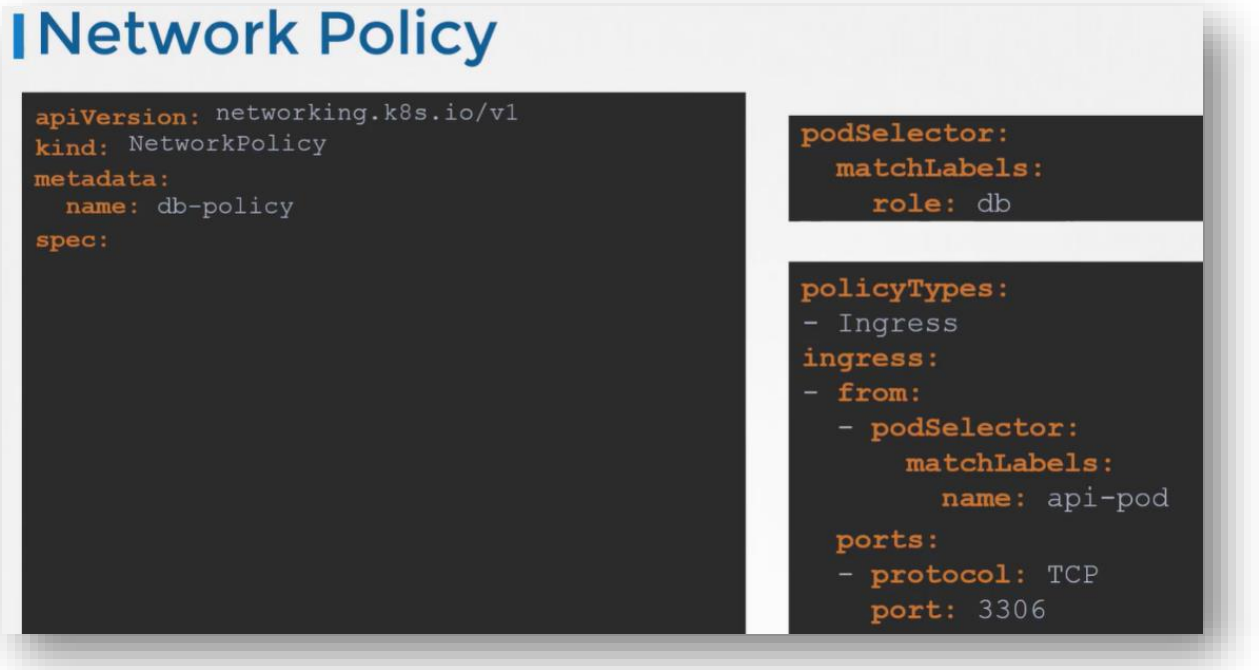
Step 5.


Network Policy
Allow Ingress Traffic From API Pod on Port 3306
DB Pod
3306
Network Policy

Step 6.


Network Policy - Rules
policyTypes:
- Ingress
ingress:
- from:
- podSelector:
matchLabels:
name: api-pod
ports:
- protocol: TCP
port: 3306
Allow
Ingress
Traffic
From
API Pod
on
Port 3306

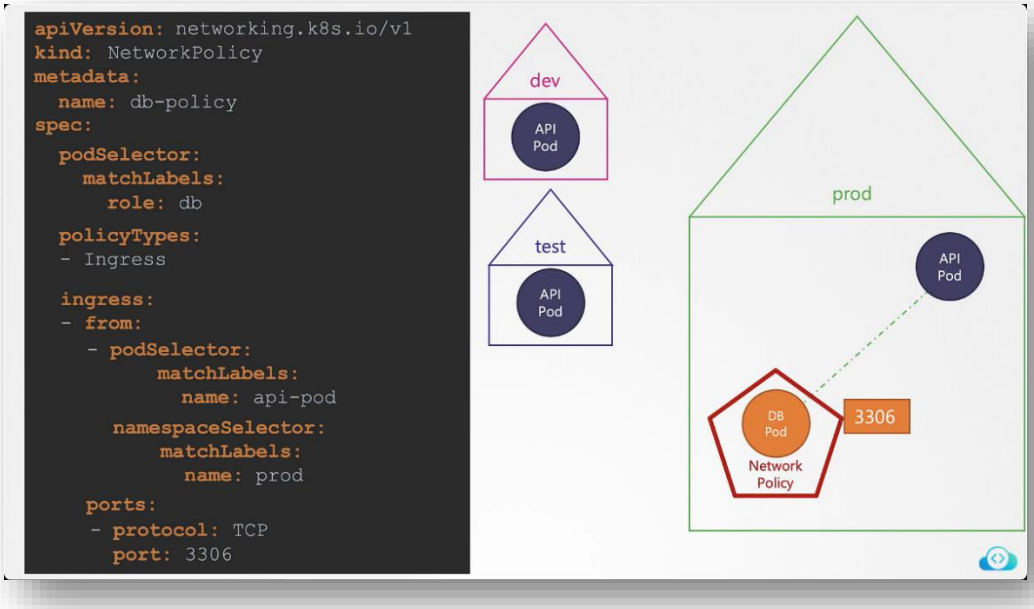
Step 7.


Network Policy
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: db-policy
spec:
podSelector:
matchLabels:
role: db
policyTypes:
- Ingress
ingress:
- from:
- podSelector:
matchLabels:
name: api-pod
ports:
- protocol: TCP
port: 3306

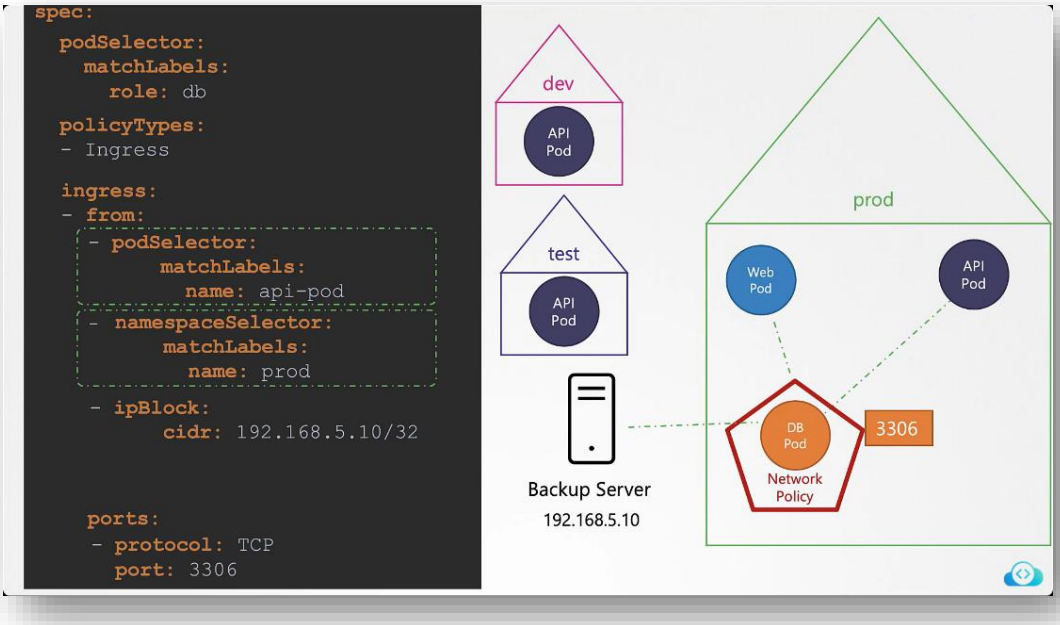
Step 8.

7.21 DEVELOPING NETWORK POLICIES

Step 1.

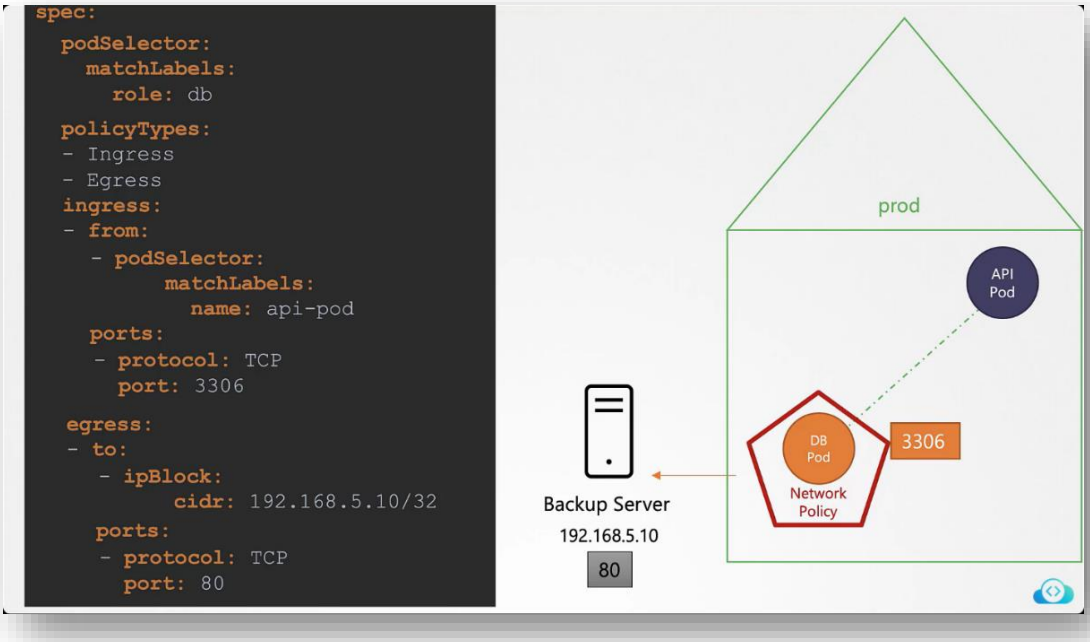


Step 2.



Step 3.

```
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          name: api-pod
    ports:
    - protocol: TCP
      port: 3306
  egress:
  - to:
    - ipBlock:
        cidr: 192.168.5.10/32
    ports:
    - protocol: TCP
      port: 80
```




```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: internal-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      name: internal
  policyTypes:
    - Egress
  egress:
    - to:
        - podSelector:
            matchLabels:
              name: payroll
      ports:
        - protocol: TCP
          port: 8080

    - to:
      - podSelector:
          matchLabels:
            name: mysql

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

7.21.1 COMMANDLINE

- kubectl get network policies
- kubectl describe netpol payroll-policy

7.22 KUBECTX AND KUBENS

- sudo git clone https://github.com/ahmetb/kubectx /opt/kubectx
- sudo ln -s /opt/kubectx/kubectx /usr/local/bin/kubectx

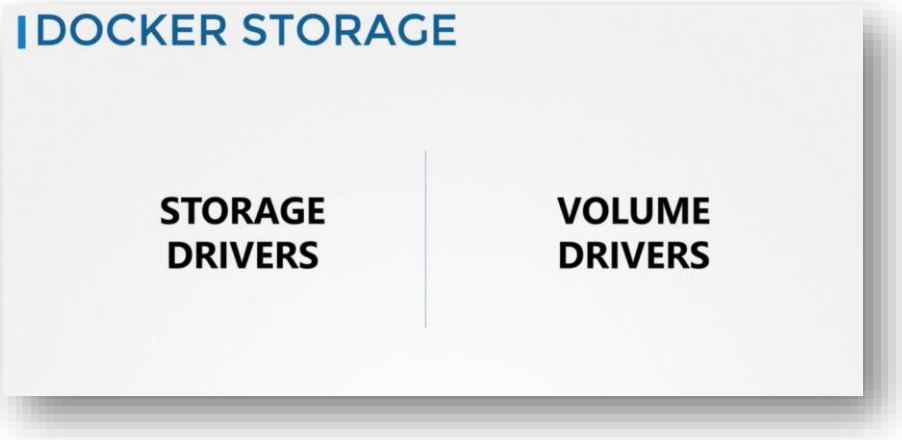
- sudo git clone https://github.com/ahmetb/kubectx /opt/kubectx
- sudo ln -s /opt/kubectx/kubens /usr/local/bin/kubens

8. STORAGE

8.1 STORAGE – SECTION INTRODUCTION

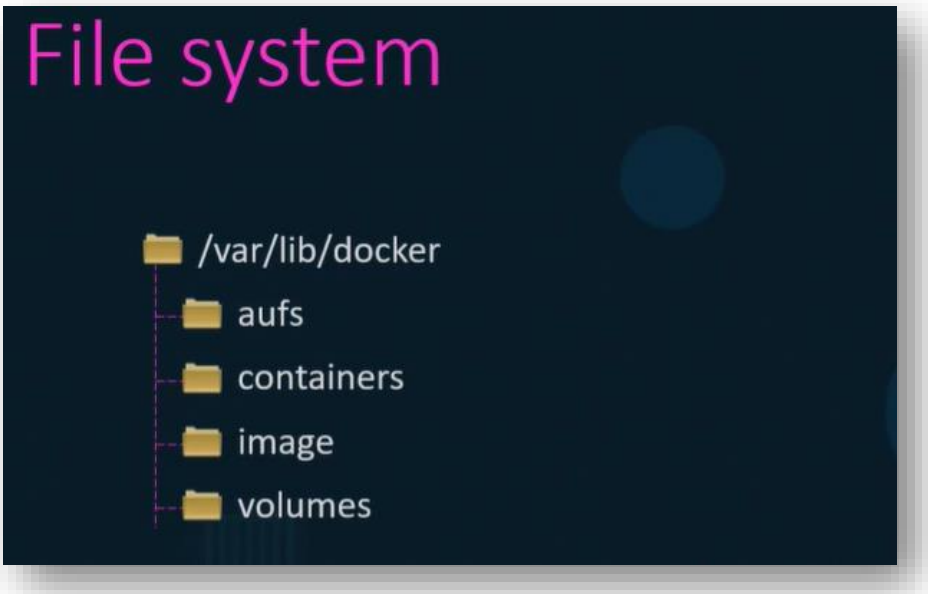
8.2 INTRODUCTION TO DOCKER STORAGE

Step 1.

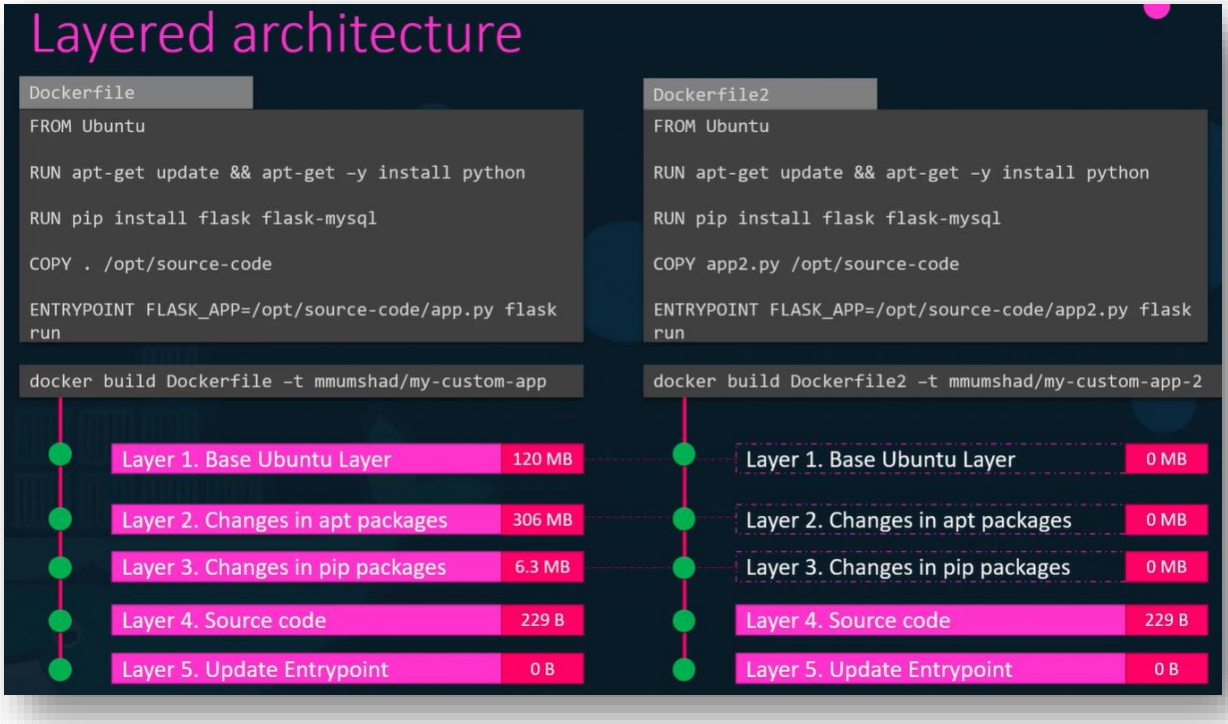


8.3 STORAGE IN DOCKER

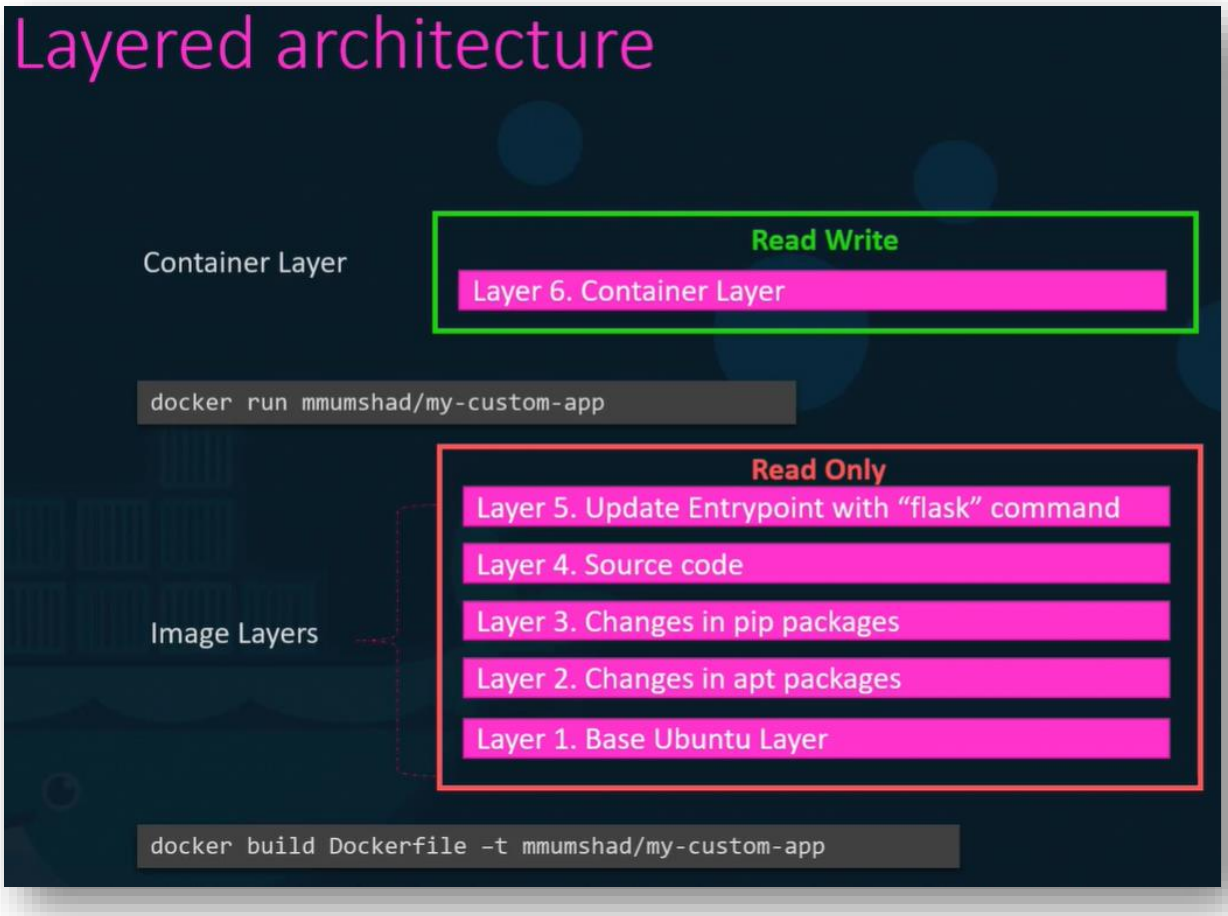
Step 1.



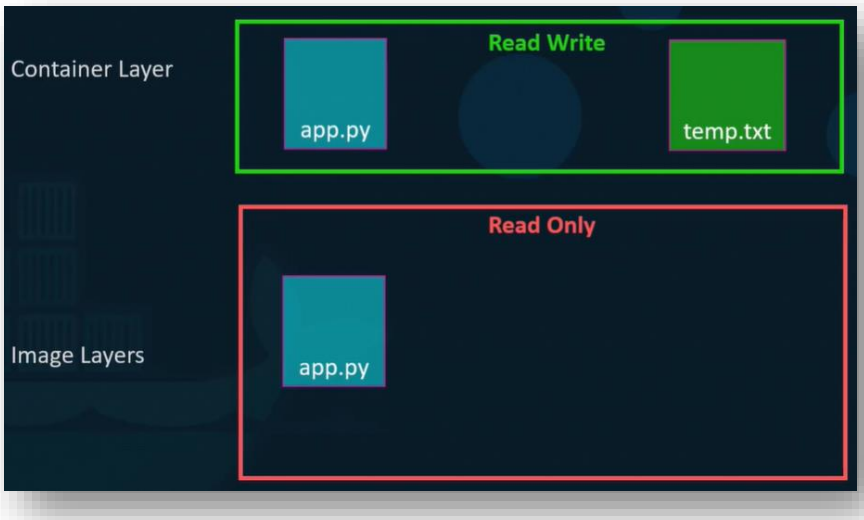
Step 2.


Layered architecture
Dockerfile
FROM Ubuntu
RUN apt-get update && apt-get -y install python
RUN pip install flask flask-mysql
COPY . /opt/source-code
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
docker build Dockerfile -t mmumshad/my-custom-app
Layer 1. Base Ubuntu Layer 120 MB
Layer 2. Changes in apt packages 306 MB
Layer 3. Changes in pip packages 6.3 MB
Layer 4. Source code 229 B
Layer 5. Update Entrypoint 0 B
Dockerfile2
FROM Ubuntu
RUN apt-get update && apt-get -y install python
RUN pip install flask flask-mysql
COPY app2.py /opt/source-code
ENTRYPOINT FLASK_APP=/opt/source-code/app2.py flask run
docker build Dockerfile2 -t mmumshad/my-custom-app-2
Layer 1. Base Ubuntu Layer 0 MB
Layer 2. Changes in apt packages 0 MB
Layer 3. Changes in pip packages 0 MB
Layer 4. Source code 229 B
Layer 5. Update Entrypoint 0 B

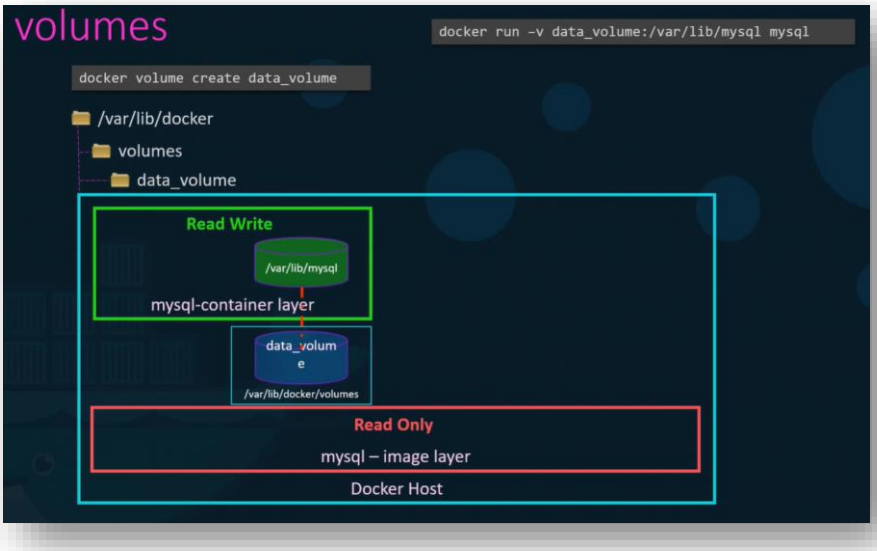
Step 3.


Layered architecture
Container Layer
Read Write
Layer 6. Container Layer
docker run mmumshad/my-custom-app
Read Only
Image Layers
Layer 5. Update Entrypoint with "flask" command
Layer 4. Source code
Layer 3. Changes in pip packages
Layer 2. Changes in apt packages
Layer 1. Base Ubuntu Layer
docker build Dockerfile -t mmumshad/my-custom-app

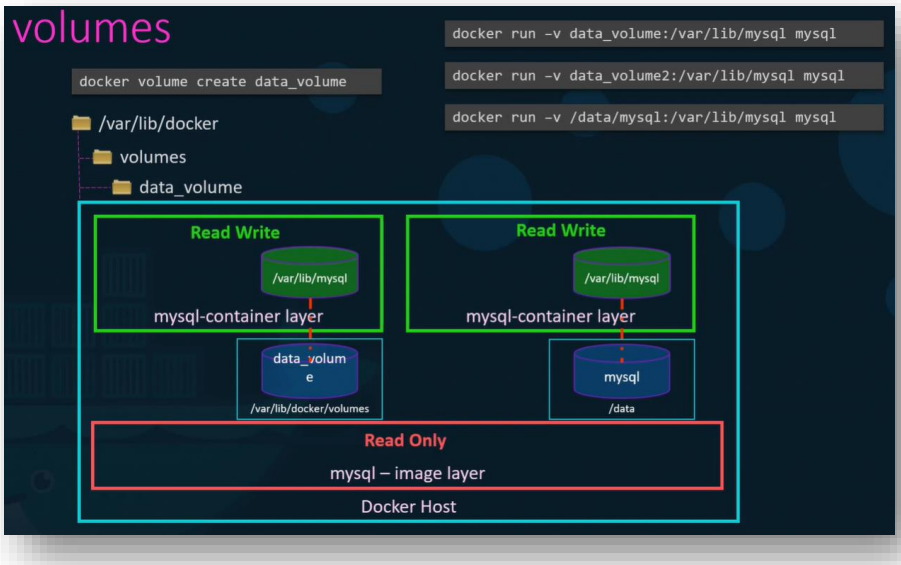
Step 4.



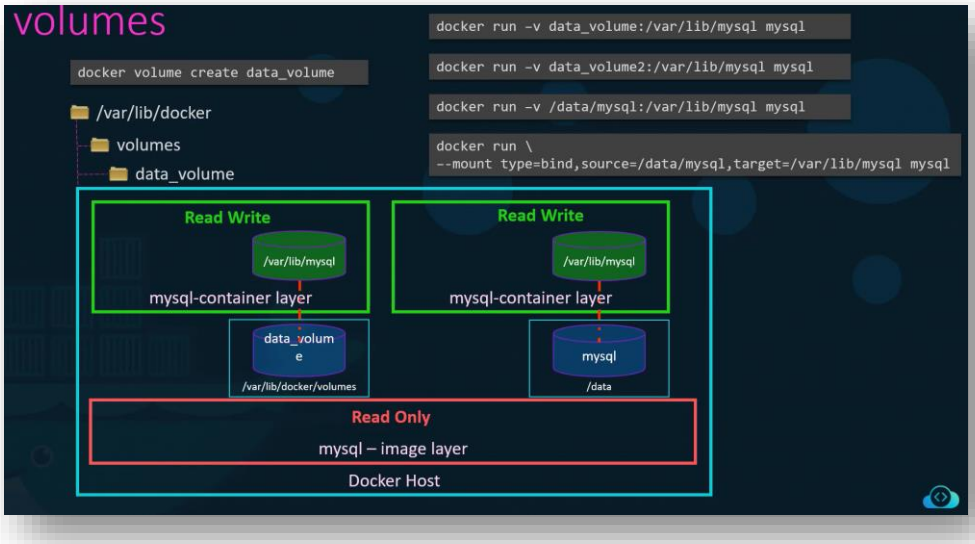
Step 5.



Step 6.



Step 7.



Step 8.

8.3.1 TYPES OF PERSISTANT VOLUME

- Volume Mounting (Mount the volume)
- Volume Binding (Mount directory or any of the locations)

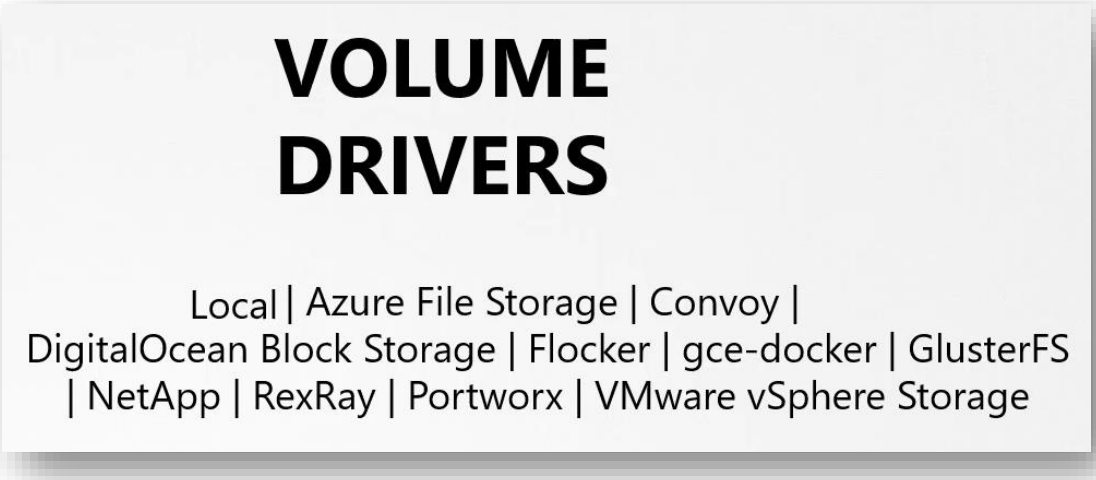
8.3.2 COMMANDLINE

- docker volume create data_volume
- docker run -v data_volume:/var/lib/mysql mysql

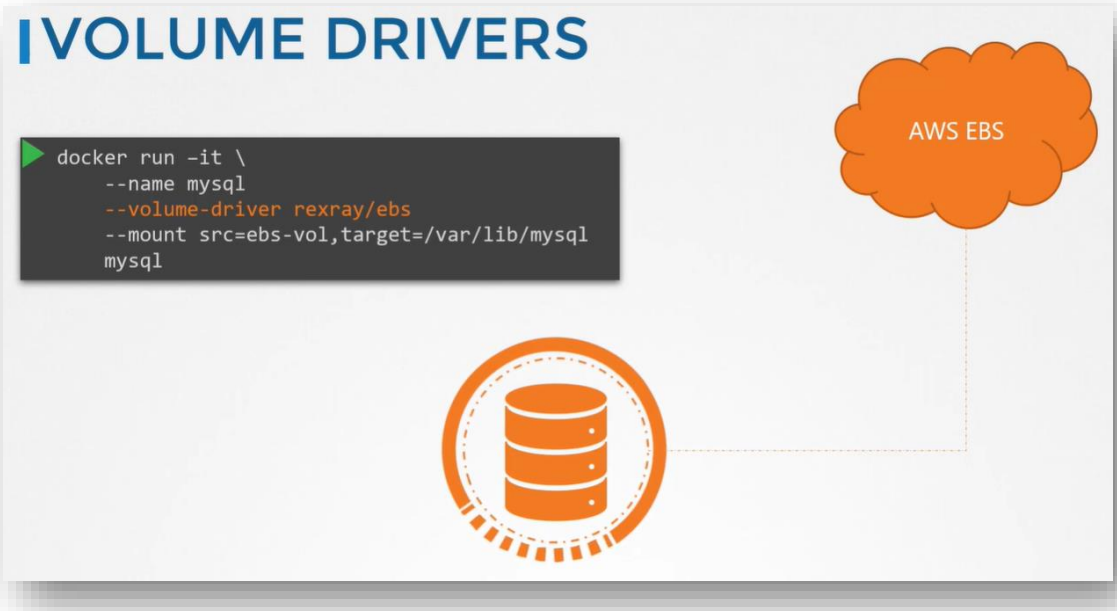
8.4 VOLUME DRIVER PLUGIN IN DOCKER

Step 1.

Step 2.

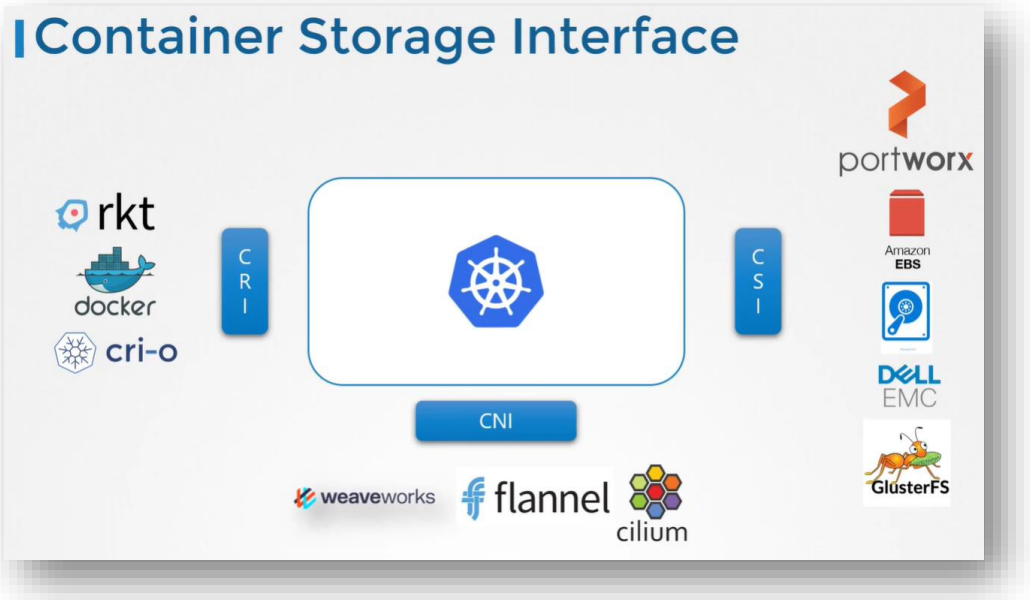


Step 3.



8.5 CONTAINER STORAGE INTERFACE

Step 1.



Step 2.

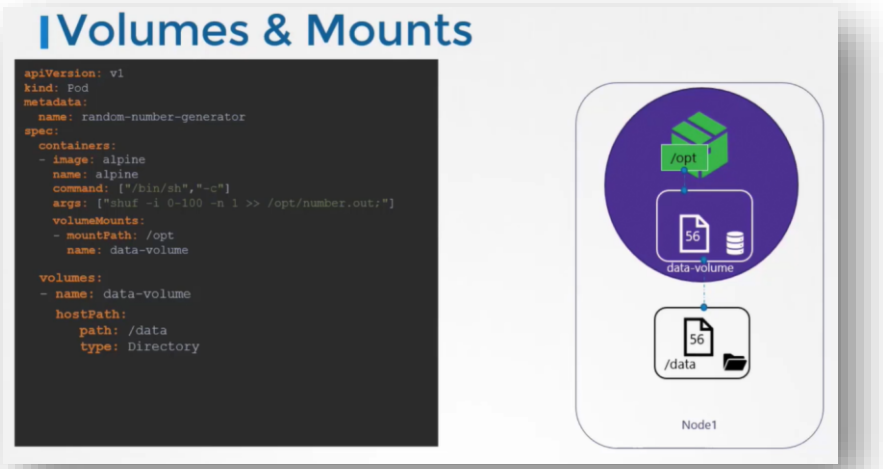
8.6 VOLUMES

Step 1.

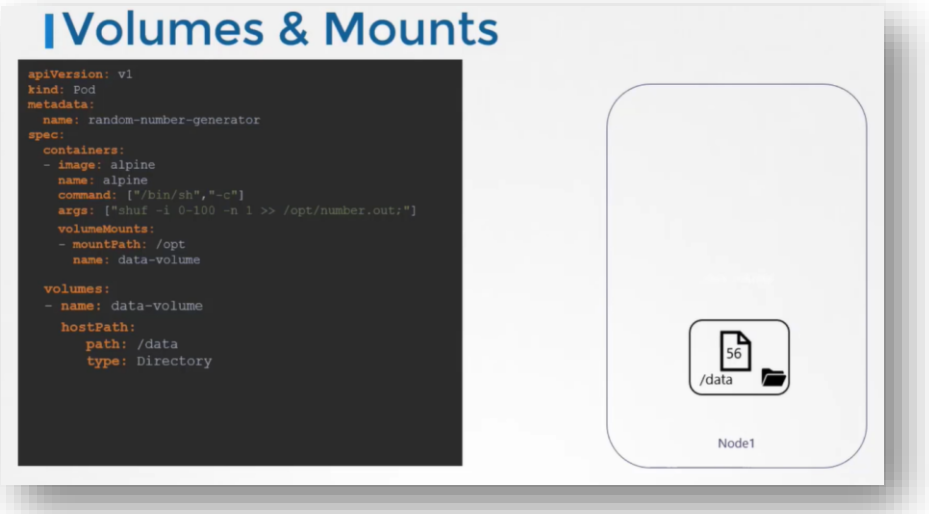
Step 2.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: random-number-generator
spec:
  containers:
  - image: alpine
    name: alpine
    command: ["/bin/sh","-c"]
    args: ["shuf -i 0-100 -n 1 >> /opt/number.out;"]
    volumeMounts:
    - mountPath: /opt
      name: data-volume

  volumes:
  - name: data-volume
    hostPath:
      path: /data
      type: Directory
```

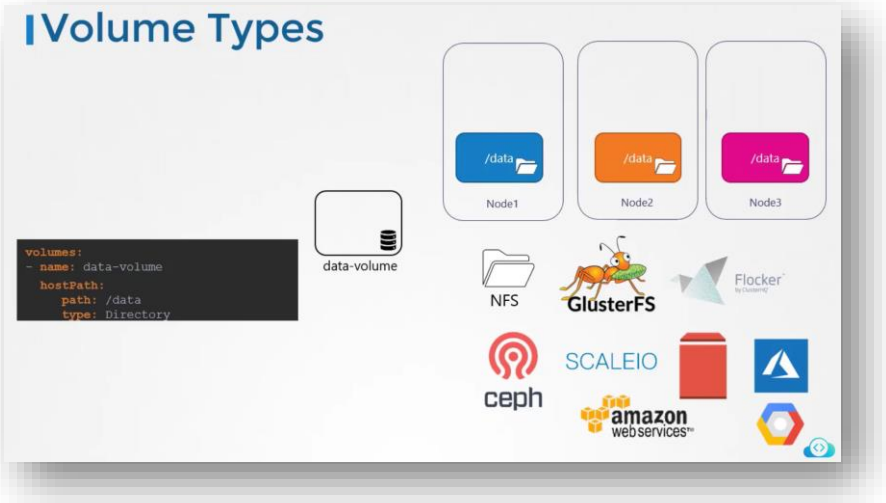
Step 3.



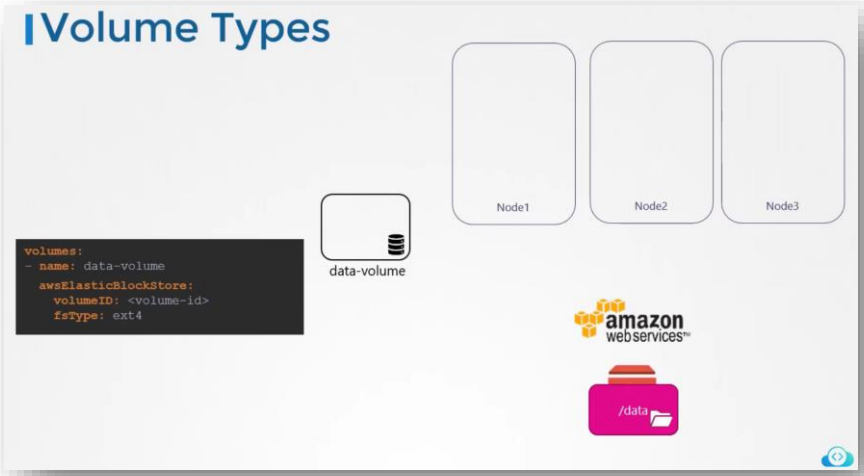
Step 4.



Step 5.

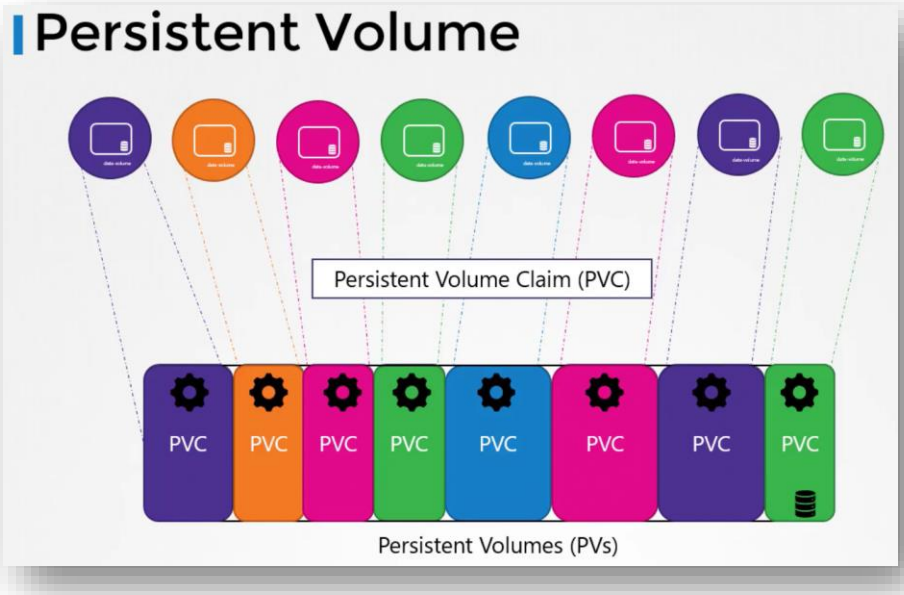


Step 6.

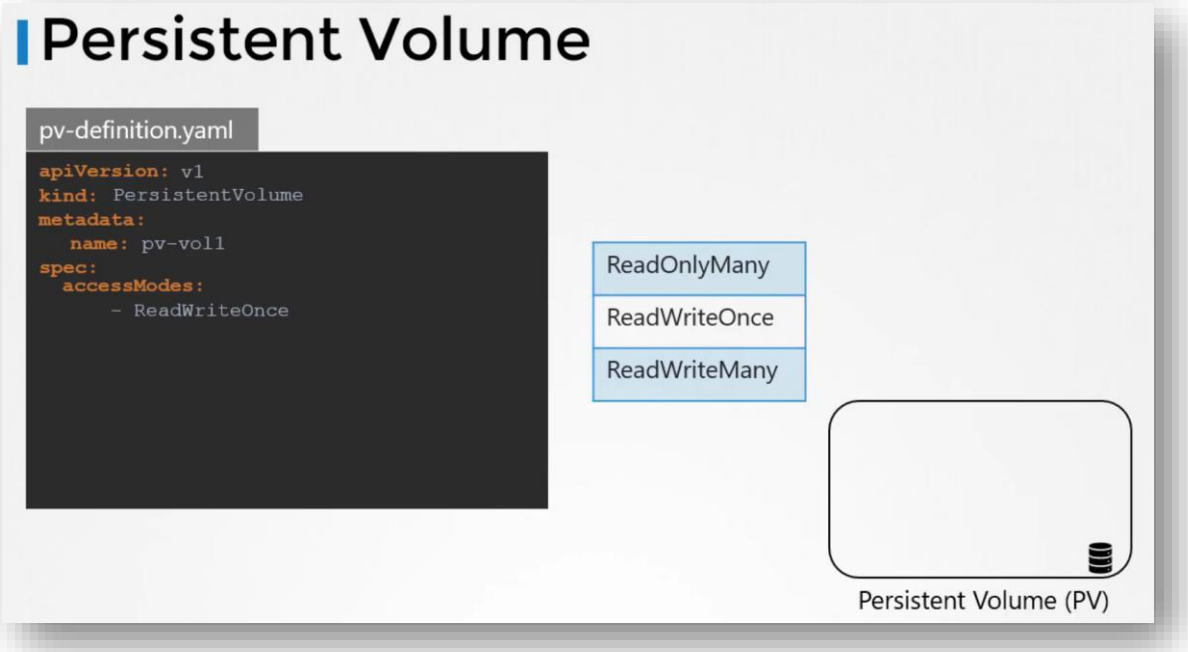


8.7 PERSISTANT VOLUMES

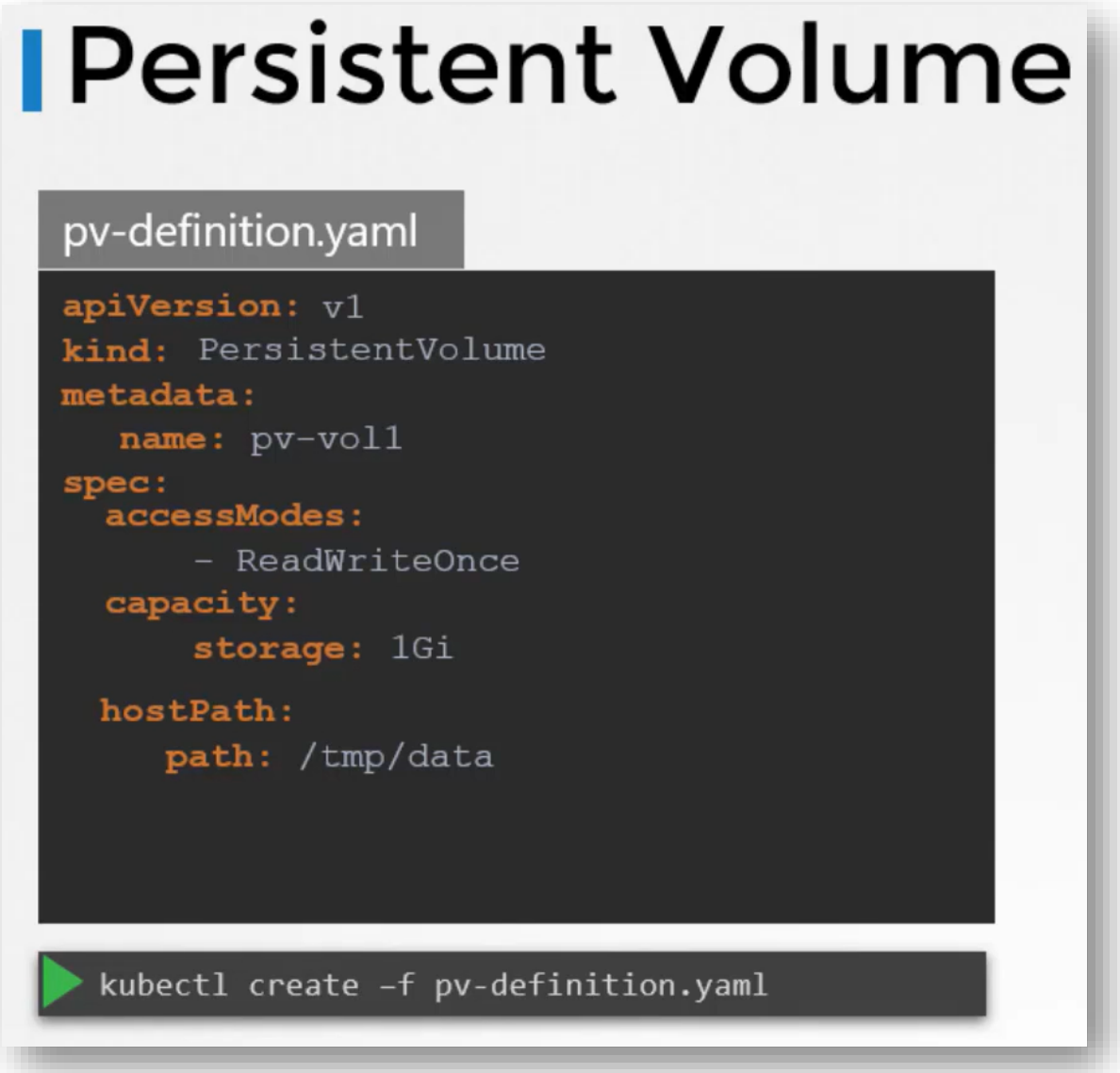
Step 1.



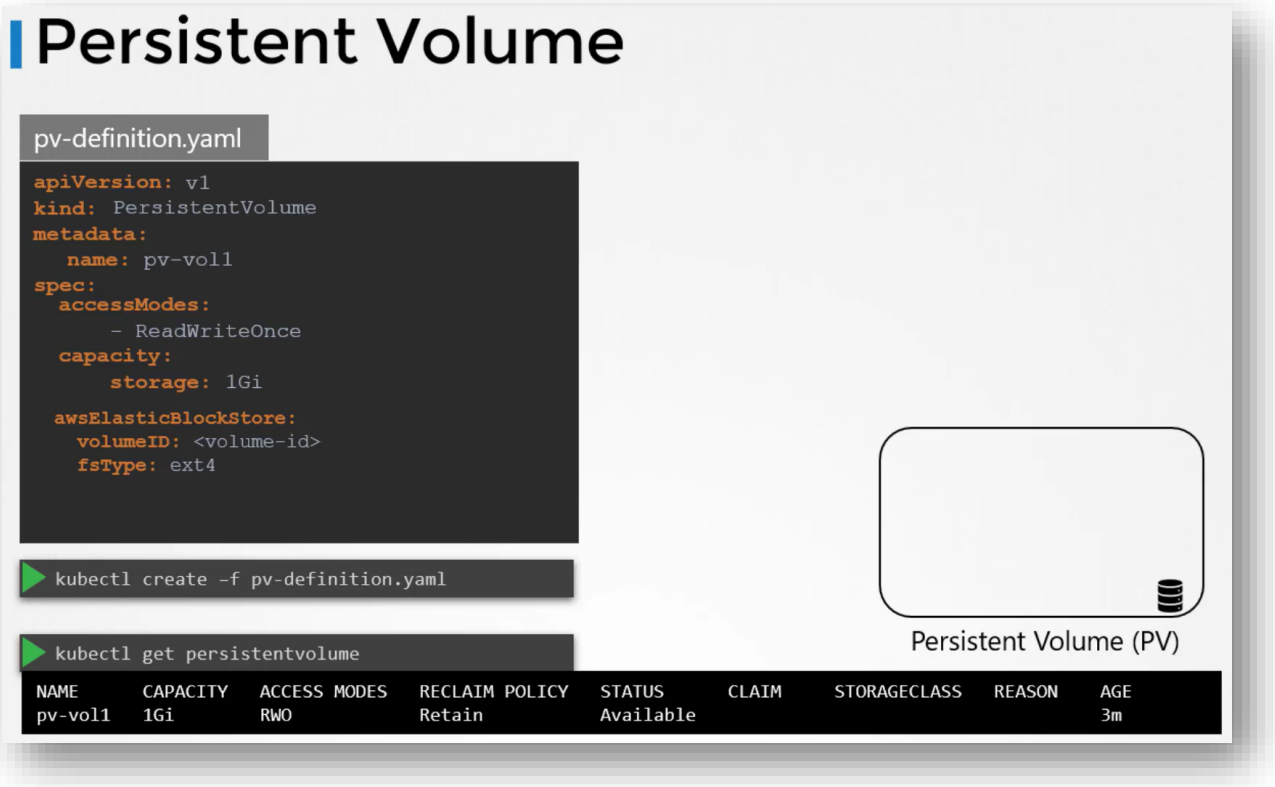
Step 2.


Persistent Volume
pv-definition.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
name: pv-vol1
spec:
accessModes:
- ReadWriteOnce
ReadOnlyMany
ReadWriteOnce
ReadWriteMany
Persistent Volume (PV)

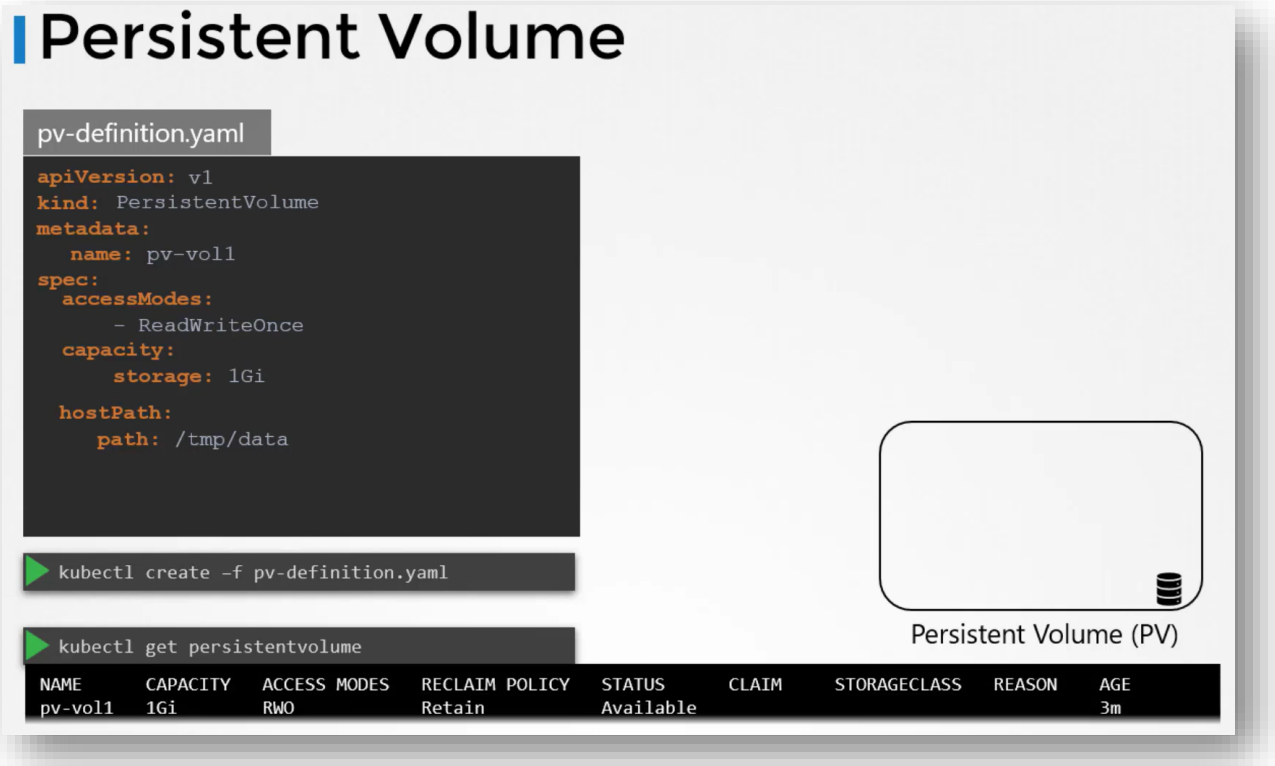
Step 3.


Persistent Volume
pv-definition.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
name: pv-vol1
spec:
accessModes:
- ReadWriteOnce
capacity:
storage: 1Gi
hostPath:
path: /tmp/data
kubectl create -f pv-definition.yaml

Step 4. Local Volume



Step 5. Cloud Storage

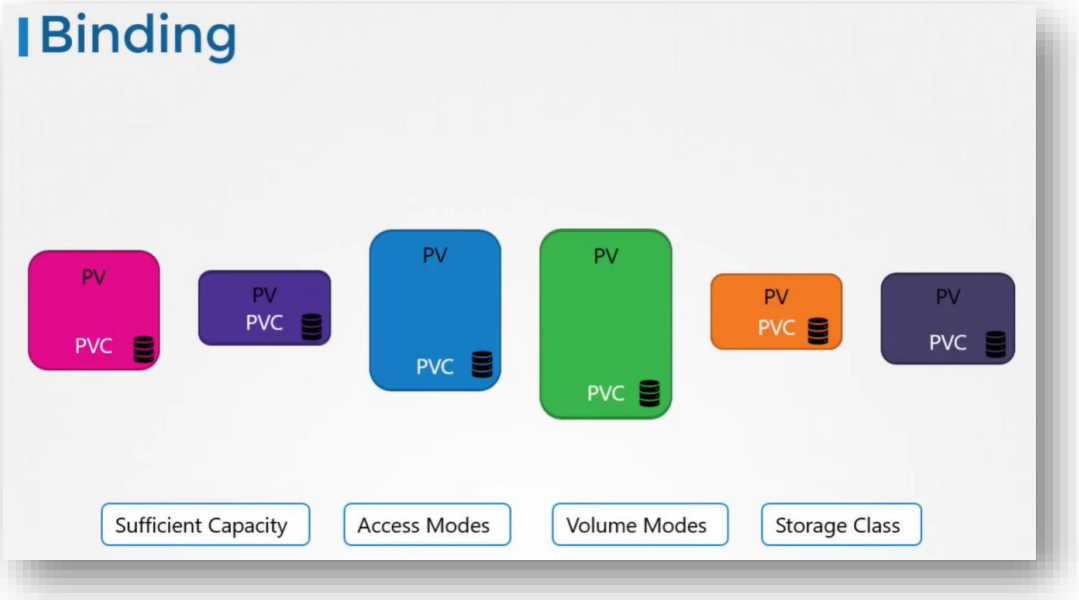


8.8 COMMANDLINE

- kubectl get pv
- kubectl describe pv pv_name

8.9 PERSISTANT VOLUME CLAIMS

Step 1.

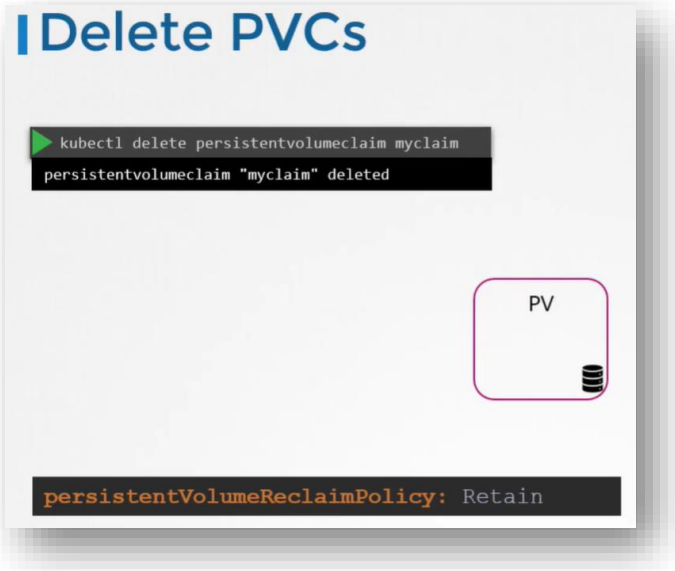


Step 2.

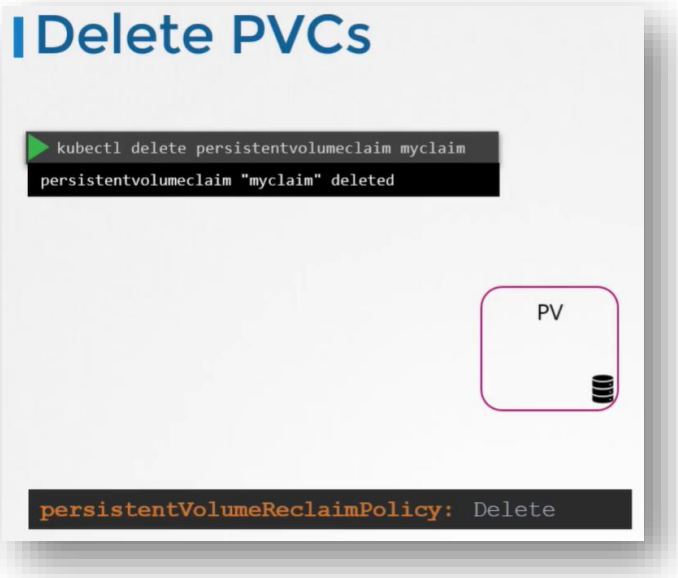
pvc-definition.yaml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500Mi
```

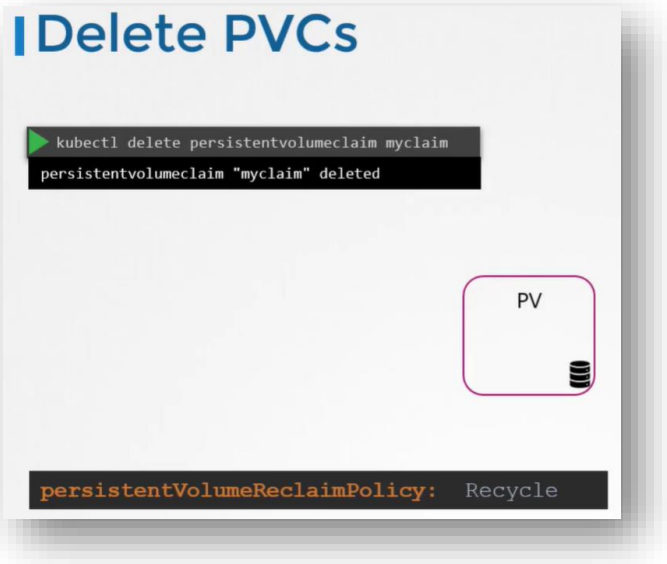
Step 3.



Step 4.



Step 5.



8.10 COMMAND LINE

- kubectl get pvc
- kubectl describe pvc pvc_name

8.11 USING PVC IN PODS

8.12 STORAGE CLASS

Step 1.

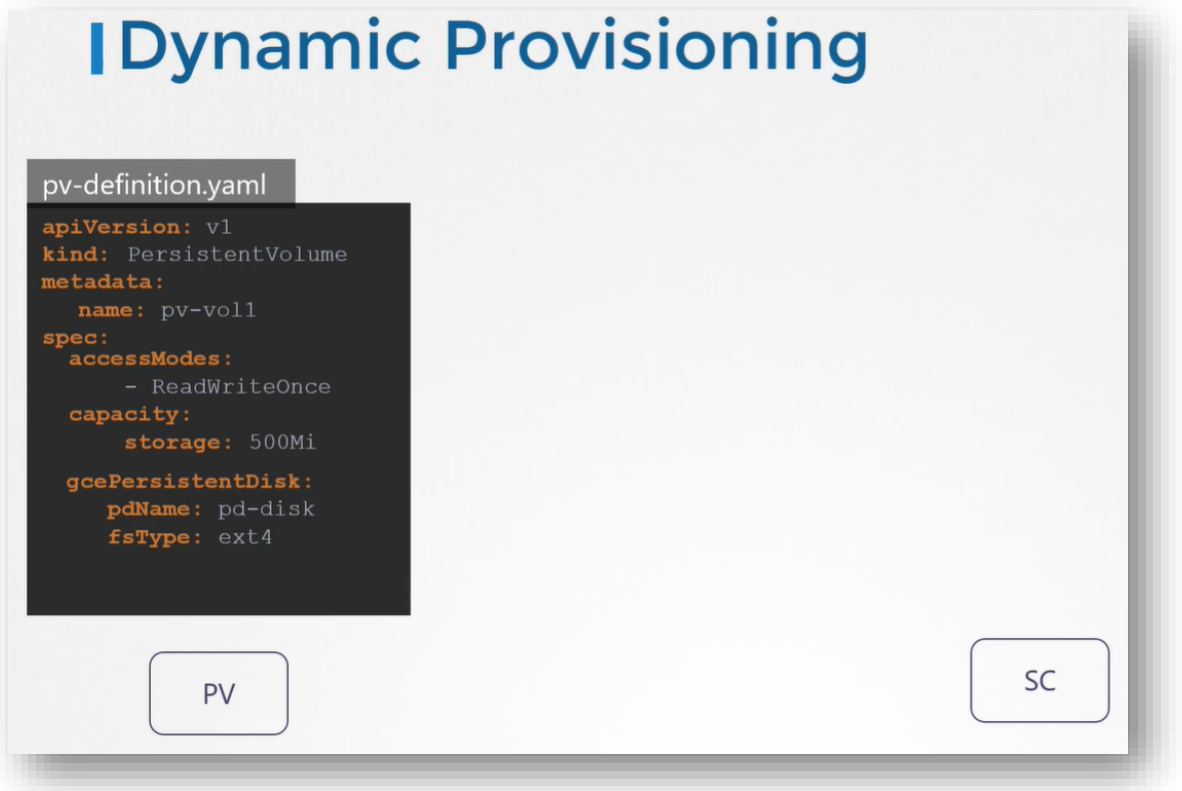
Step 2.

Step 3.

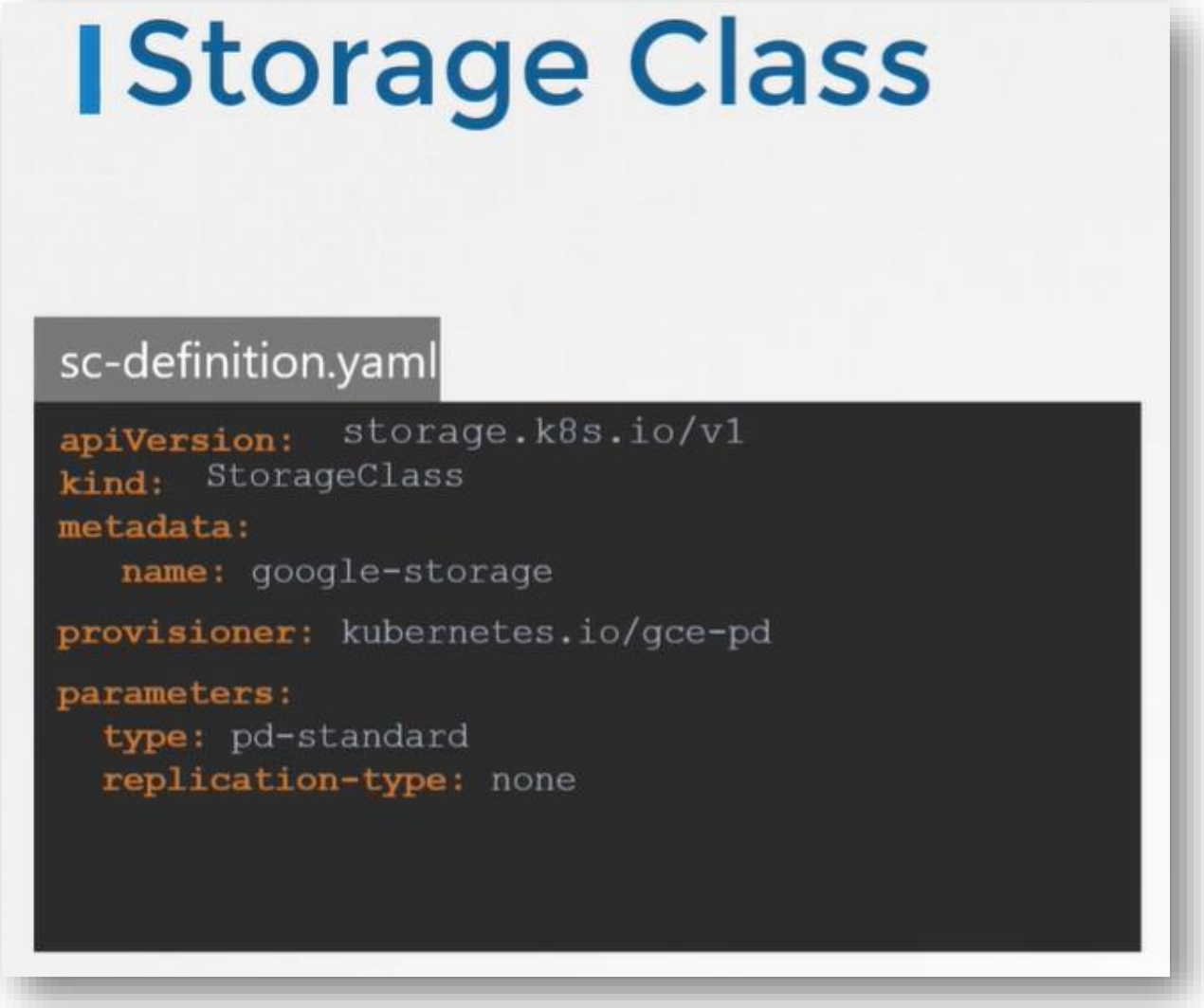
pv-definition.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
   name: pv-vol1
spec:
  accessModes:
      - ReadWriteOnce
  capacity:
      storage: 500Mi
  gcePersistentDisk:
     pdName: pd-disk
     fsType: ext4
```

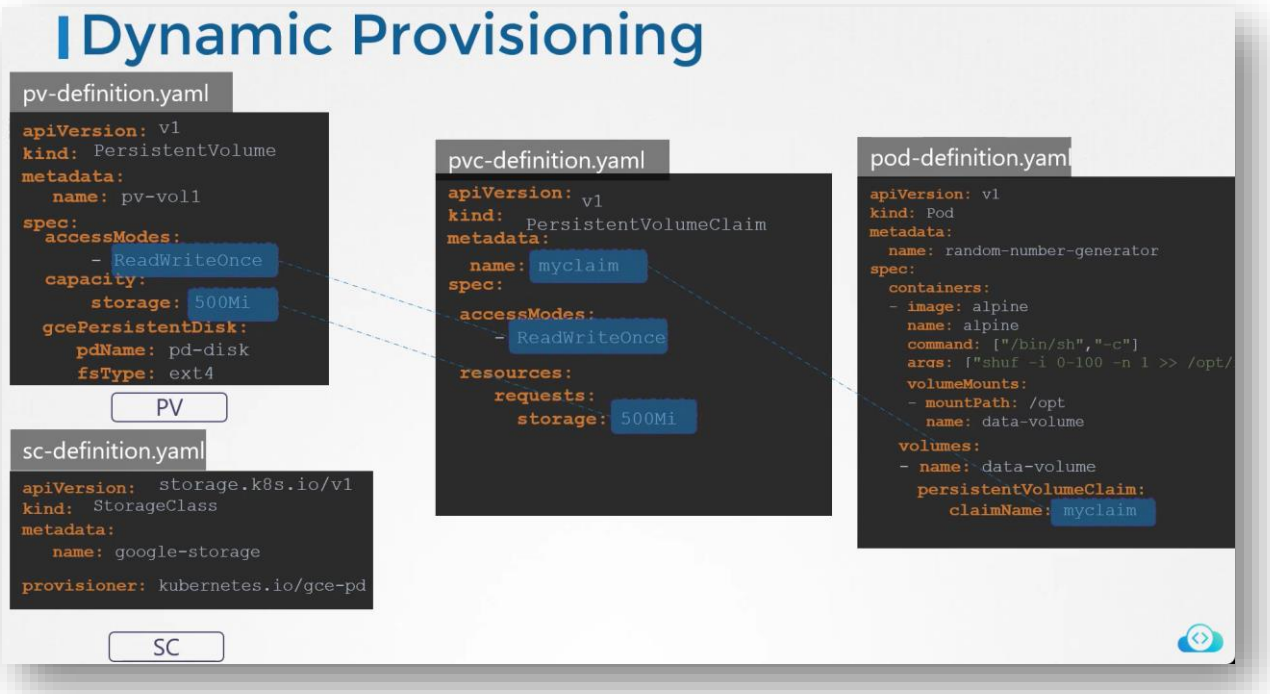
Step 4.



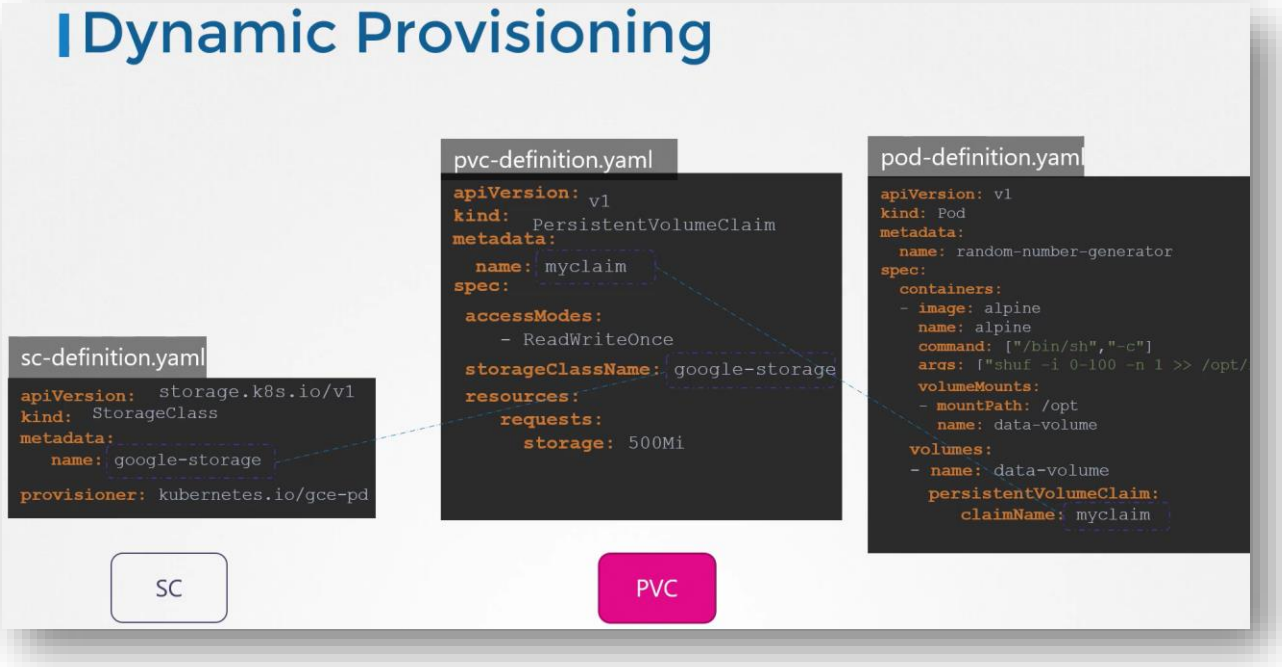
Step 5.


Storage Class
sc-definition.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: google-storage
provisioner: kubernetes.io/gce-pd
parameters:
type: pd-standard
replication-type: none

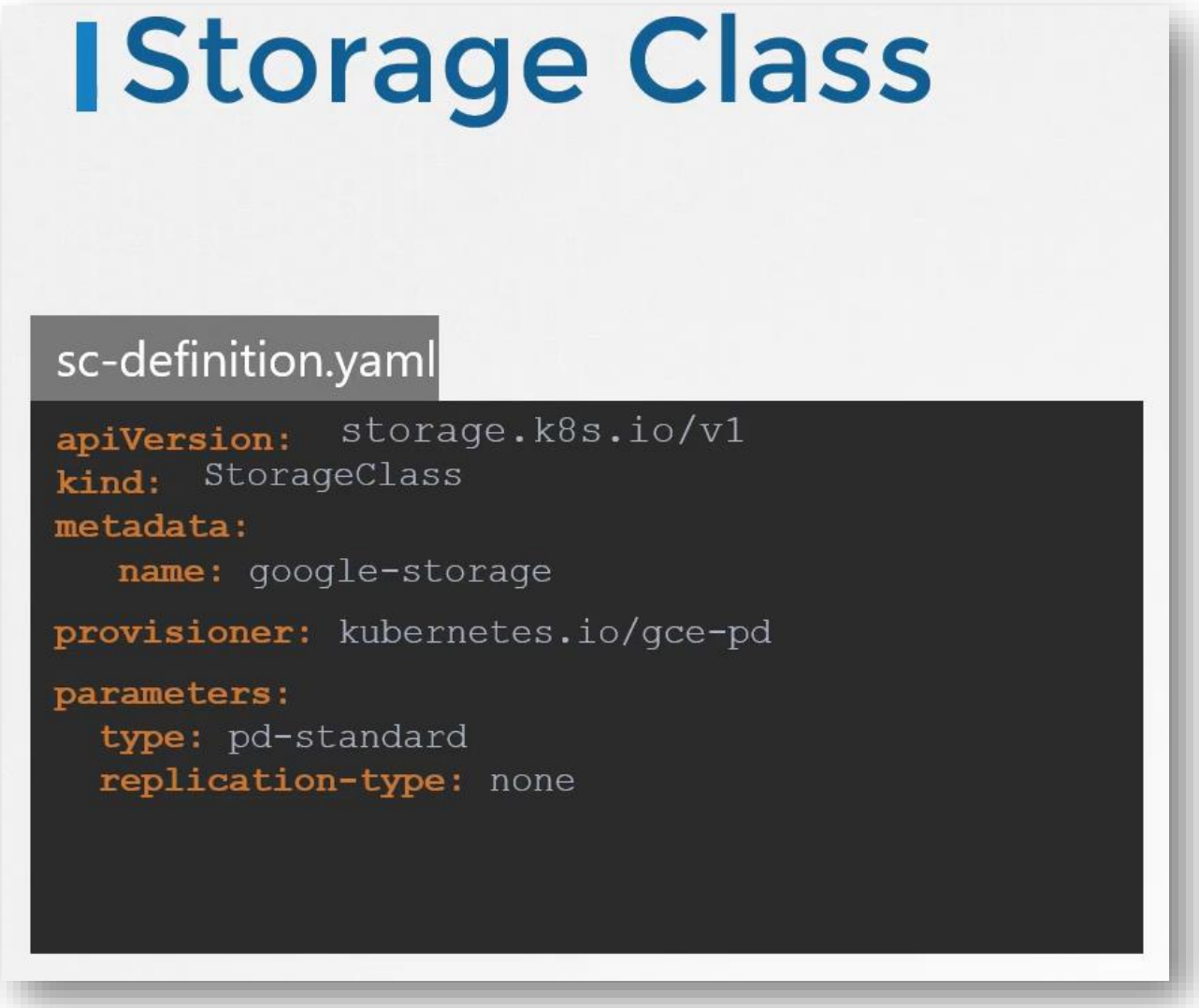
Step 6.


Dynamic Provisioning
pv-definition.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
name: pv-vol1
spec:
accessModes:
- ReadWriteOnce
capacity:
storage: 500Mi
gcePersistentDisk:
pdName: pd-disk
fsType: ext4
PV
sc-definition.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: google-storage
provisioner: kubernetes.io/gce-pd
SC
pvc-definition.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: myclaim
spec:
accessModes:
- ReadWriteOnce
resources:
requests:
storage: 500Mi
pod-definition.yaml
apiVersion: v1
kind: Pod
metadata:
name: random-number-generator
spec:
containers:
- image: alpine
name: alpine
command: ["/bin/sh","-c"]
args: ["shuf -i 0-100 -n 1 >> /opt/
volumeMounts:
- mountPath: /opt
name: data-volume
volumes:
- name: data-volume
persistentVolumeClaim:
claimName: myclaim

Step 7.

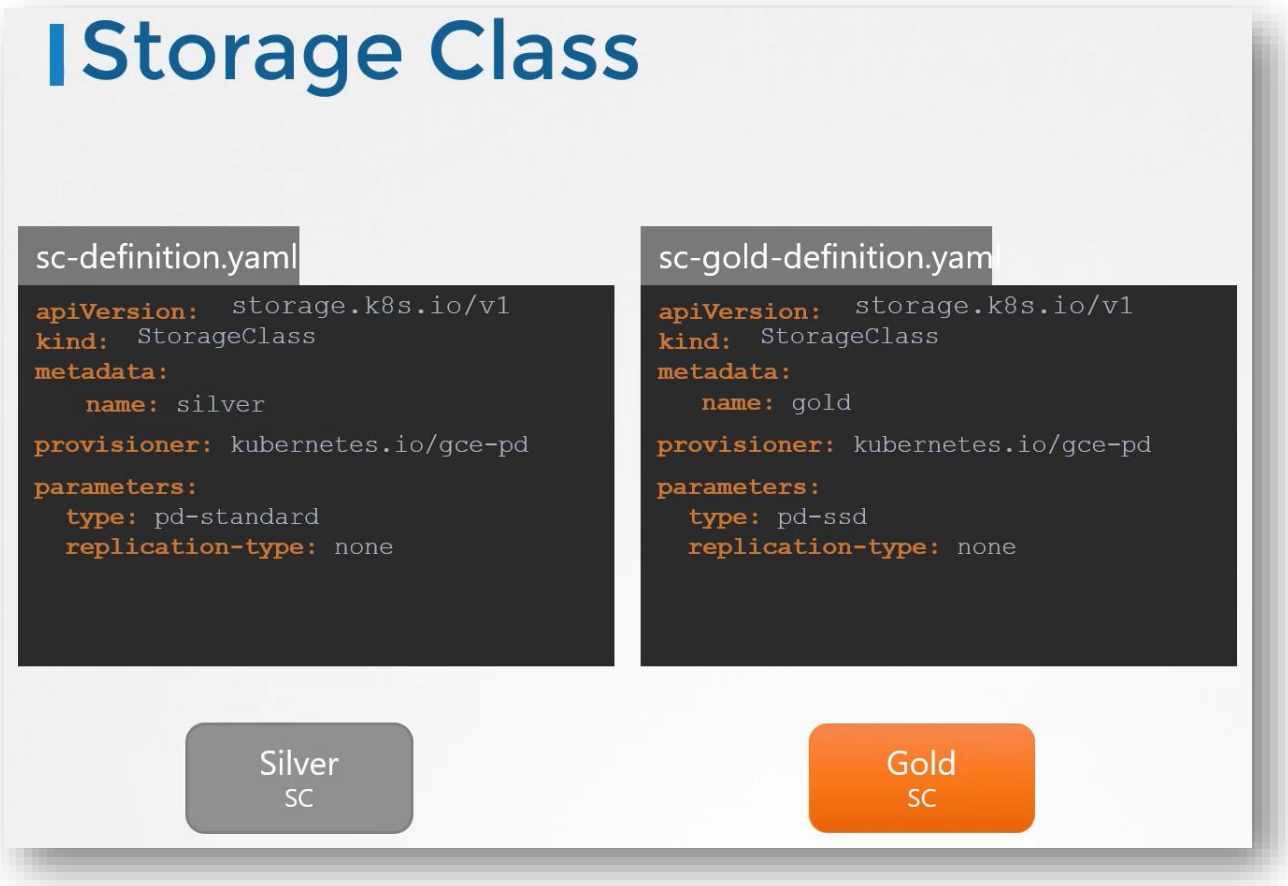

Dynamic Provisioning
pvc-definition.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: myclaim
spec:
accessModes:
- ReadWriteOnce
storageClassName: google-storage
resources:
requests:
storage: 500Mi
pod-definition.yaml
apiVersion: v1
kind: Pod
metadata:
name: random-number-generator
spec:
containers:
- image: alpine
name: alpine
command: ["/bin/sh","-c"]
args: ["shuf -i 0-100 -n 1 >> /opt/
volumeMounts:
- mountPath: /opt
name: data-volume
volumes:
- name: data-volume
persistentVolumeClaim:
claimName: myclaim
sc-definition.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: google-storage
provisioner: kubernetes.io/gce-pd
SC
PVC

Step 8.


Storage Class
sc-definition.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: google-storage
provisioner: kubernetes.io/gce-pd
parameters:
type: pd-standard
replication-type: none

Step 9.

Volume Plugin
AWSElasticBlockStore
AzureFile
AzureDisk
CephFS
Cinder
FC
FlexVolume
Flocker
GCEPersistentDisk
Glusterfs
iSCSI
Quobyte
NFS
RBD
VsphereVolume
PortworxVolume
ScaleIO
StorageOS



8.13 COMMANDLINE

- kubectl get sc
- kubectl describe sc
- kubectl describe sc local-storage
- kubectl get pvc
- kubectl describe pvc

9. NETWORKING

9.1 COMMANDLINE

- ip a Or ip link
- ip link show eth0
- ip route show default
- netstat -nplt
- kubectl get po -owide -n kube-system | grep weave
- kubectl get configmap
- kubectl describe configmap coredns -n kube-system

10. DESIGN AND INSTALL A KUBERNETES CLUSTER

11. INSTALL KUBERNETES THE KUBEADM WAY

12. TROUBLESHOOTING

12.1 APPLICATION FAILURE

12.2 CONTROL PLAN FAILURE

12.2.1 COMMANDLINE

- kubectl get nodes
- kubectl get pods
- kubectl get pods -n kube-system
- service kube-apiserver status
- service kube-controller-manager status
- kubectl logs -n kube-system kube-controller-manager-controlplane
- services kube-scheduler status
- service kubelet status
- service kube-proxy status
- kubectl logs kube-apiserver-master -n kube-system
- sudo Journalctl -u kube-apiserver
- source <(kubectl completion bash)

13. KUBERNETES FUNDAMENTALS

- **Kubernetes**: An open-source platform for automating the deployment, scaling, and operations of application containers.

Node: A physical or virtual machine serving as a worker in Kubernetes.

Pod: The smallest deployable unit in Kubernetes, representing one or more containers.

Container: A portable, self-sufficient software package that includes all necessary components to run an application.

ReplicaSet: Ensures a specified number of pod replicas are running at any given time.

Deployment: Manages ReplicaSets and provides declarative updates to Pods and ReplicaSets.

Service: Defines a logical set of Pods and a policy for accessing them.

Ingress: Manages external access to the services in a cluster, typically for HTTP.

Namespace: Divides cluster resources between multiple users, enabling resource quota management.

Volume: A directory containing data accessible to containers in a Pod.

PersistentVolume (PV): Storage in the cluster provisioned by an administrator or dynamically using Storage Classes.

PersistentVolumeClaim (PVC): A user's request for storage.

StatefulSet: Manages the deployment and scaling of a set of Pods with guarantees about ordering and uniqueness.

ConfigMap: Stores non-confidential data in key-value pairs.

Secret: Stores and manages sensitive information such as passwords and tokens.

DaemonSet: Ensures all (or some) Nodes run a copy of a Pod.

Job: Manages the completion of a specific task or batch job in the cluster.

14. KUBERNETES BASIC COMMAND

- kubectl get nodes

- kubectl get deployment
- kubectl get pods
- kubectl get services
- kubectl describe deployment deployment-1
- kubectl get pods --output=wide (For check pods IP)
- kubectl get replicasets (For check replicas)
- kubectl delete pods -all (Delete pods)

15. KUBENATES IMPERATIVES COMMANDS

- kubectl run nginx --image=nginx:latest --dry-run=client -o yaml > task.yaml (To Create YAML file)
- kubectl run pod-1 --image=nginx:latest (Create a Pod)
- kubectl create deployment deployment-1 --image=nginx:latest (To Create Deployment)
- kubectl expose deployment deployment-1 --port 80 (To Create Service)
- kubectl scale deployment deployment-1 --replicas=3 (To Create Replicas)
- kubectl edit deployment deployment-1 (To Edit Deployment)
- kubectl set image deployment deployment-1 nginx=nginx:1.18 (To Change Image)
- kubectl describe deployment deployment-1 (To Describe the details)
- kubectl create deployment deployment-1 --image=nginx:latest --replicas=3

16. KUBERTNETES CLUSTER SETUP ON UBUNTU

All Nodes

```
sudo apt update
sudo apt upgrade
sudo apt install net-tools
sudo hostnamectl set-hostname MasterNode01
sudo vim /etc/hosts
sudo swapoff -a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
sudo vim /etc/modules-load.d/containerd.conf
```

overlay
br_netfilter

```
sudo modprobe overlay
sudo modprobe br_netfilter
sudo vim /etc/sysctl.d/kubernetes.conf
```

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
```

```
sudo sysctl --system
```

sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

sudo apt update

sudo apt install -y containerd.io

sudo containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1

sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

sudo systemctl restart containerd

sudo systemctl enable containerd

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

sudo apt-get update

sudo apt install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

Master Node Only

sudo kubeadm init --apiserver-advertise-address=10.251.0.11 --pod-network-cidr=10.0.0.0/8 --ignore-preflight-errors=all

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml

---

WorkerNode Only

kubeadm join 192.168.251.8:6443 --token k6enj0.egkshbevkiynvgnt \ --discovery-token-ca-cert-hash sha256:063117edb34c465fd0a6328312abb3199c60b19ecaf5374b792b0dd4709de1a1s

kubeadm join 10.251.0.11:6443 --token roatlj.wphuue848fh9oo40 \ --discovery-token-ca-cert-hash sha256:4c5754e1720080f914ab033a5a4e57637de375d3b4d096c83e366c6811d453d6

sudo kubeadm join 10.251.0.11:6443 --token skc5tf.nma43iomkyhh1mua \ --discovery-token-ca-cert-hash sha256:4c5754e1720080f914ab033a5a4e57637de375d3b4d096c83e366c6811d453d6

17.

All Nodes

```
sudo apt update
sudo apt upgrade
sudo apt install net-tools
sudo hostnamectl set-hostname MasterNode01
sudo vim /etc/hosts
sudo swapoff -a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab

sudo tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF

sudo modprobe overlay
```

```
sudo modprobe br_netfilter

sudo tee /etc/sysctl.d/kubernetes.conf <<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF

sudo sysctl --system

sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o
/etc/apt/trusted.gpg.d/docker.gpg

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -
cs) stable"

sudo apt update

sudo apt install -y containerd.io


sudo containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1

sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml


sudo systemctl restart containerd

sudo systemctl enable containerd


echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg

sudo apt-get update

sudo apt install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl
```

```
sudo apt update

sudo apt upgrade

sudo apt install net-tools

sudo hostnamectl set-hostname masternode01

sudo tee /etc/hosts <<EOF

10.251.0.11     masternode01

EOF

sudo swapoff -a

sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab

sudo usermod -aG root myadmin

sudo su

sudo tee /etc/modules-load.d/containerd.conf <<EOF

overlay

br_netfilter

EOF

sudo modprobe overlay

sudo modprobe br_netfilter

sudo tee /etc/sysctl.d/kubernetes.conf <<EOF

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip_forward = 1

EOF

sudo sysctl --system

sudo apt install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable

"enter"

sudo apt update

sudo apt install -y containerd.io

sudo containerd –version

sudo systemctl restart containerd

sudo systemctl enable containerd

sudo apt update

sudo apt list --upgradable

sudo containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1

sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

sudo apt-get update

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

sudo apt-get update

sudo apt install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

sudo systemctl enable kubelet

sudo systemctl start kubelet

sudo systemctl status kubelet

kubeadm version

sudo systemctl restart containerd

sudo systemctl enable containerd

sudo apt update

sudo kubeadm init --apiserver-advertise-address=192.168.251.10 --pod-network-cidr=192.168.0.0/16 --ignore-preflight-errors=all

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml

19.