

KUBERNETES CONFIGURATIONS

Author: Mehul Solanki

Date: 13-July-2024

Version: 1.0

Document History

<i>Author</i>	<i>Date</i>	<i>Comments</i>
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

1. CONTENTS.....	8
1.1 Core Concepts	8
1.2 Scheduling.....	8
1.3 Logging Monitoring.....	8
1.4 Applications Lifecycle Management	8
1.5 Cluster Maintenance.....	8
1.6 Security	8
1.7 Storage	8
1.8 Networking.....	8
1.9 Installation, Configuration & Validation	8
1.10 Troubleshooting.....	8
2. CORE CONCEPTS.....	9
2.1 CLUSTER ACHITECTURE.....	9
2.1.1 MASTERNODES (Manage, Plan, Schedule, Monitor, Nodes)	10
2.1.2 WORKER NODES (Host Applications as Container).....	10
2.2 DOCKER-VS-CONTAINER-D.....	10
2.2.1	10
2.2.2	11
2.3 ETCD	11
2.3.1 ETCD-COMMANDS	12
2.4 KUBE-API SERVER.....	13
2.4.1	13
2.4.2	13
2.5 KUBECONTROL MANAGER.....	13
2.6 KUBE SCHEDULER.....	14
2.7 KUBELET	14
2.8 KUBEPROXY.....	15
2.9 PODS	16
2.9.1 COMPONENETS.....	16
2.9.2 PODS STRUCTURE	16
2.9.3 COMMANDLINE	16
2.10 MULTI CONTAINER IN A SINGLE POD.....	17
2.10.1 COMMANDLINE	17
2.11 POD YML STRUCTURE	18
2.11.1 COMMAND LINE	18
2.11.2 BASIC YML FILE.....	18

2.12	REPLICASET	18
2.12.1	19
2.12.2	COMMANDLINE	19
2.13	DEPLOYMENTS	19
2.13.1	COMPONENTS.....	19
2.13.2	20
2.13.3	20
2.13.4	20
2.13.5	COMMAND LINE	20
2.14	SERVICES	21
2.14.1	TYPES OF SERVICES	21
2.14.2	COMMANDLINE	22
2.14.3	CLUSTER IP	22
2.14.4	COMMANDLINE	22
2.14.5	LOAD BALANCER	23
2.14.6	COMMANDLINE	24
2.15	NAMESPACE	24
2.15.1	AVAILABLE NAMESPACES BY DEFAULT	24
2.15.2	YML FILE	24
2.15.3	COMMADLINE	24
2.16	IMPERATIVE V/S DECLARATIVE	25
2.16.1	COMMANDLINE	25
3.	<i>SCHEDULING</i>	25
3.1	MANUAL SCHEDULING.....	25
3.1.1	COMMAND LINE	26
3.2	LABELS AND SELECTORS.....	26
3.2.1	COMMANDLINE	26
3.3	TAINTS AND TOLERATIONS	27
3.3.1	COMMANDLINE	28
3.4	NODE SELECTOR.....	28
3.4.1	COMMANDLINE	29
3.5	NODE AFFINITY	29
3.5.1	TYPE.....	29
3.6	TAINTS AND TOLERATIONS V/S NODE AFFINITY.....	31
3.7	RESOURCE LIMIT	33
3.8	DAEMON SET	36
3.8.1	COMMAND LINE	36

3.9	STATIC PODS	37
3.9.1	COMMANDLINE	38
3.10	MULTIPLE SCHEDULERS	38
3.10.1	COMMANDLINE	40
4.	LOGGING AND MONITORING.....	40
4.1	MONITOR CLUSTER COMPONENTS	40
4.1.1	COMMANDLINE	40
4.2	MANAGING APPLICATIONS LOGS	40
4.2.1	COMMANDLINE	41
5.	APPLICATIONS LIFECYCLE MANAGEMENT.....	41
5.1	ROLLING UPDATES AND ROLLBACKS	41
5.2	COMMANDS AND ARGUMENTS IN DOCKER.....	44
5.3	COMMANDS AND ARGUMENTS IN KUBERNETES.....	44
5.3.1	COMMANDLINE	46
5.4	CONFIGURE ENVIRONMENT VARIABLES IN APPLICATIONS.....	46
5.5	CONFIGURE CONFIG MAP IN APPLICATIONS.....	47
5.5.1	COMMANDLINE	49
5.6	SECRETS.....	49
5.6.1	COMMANDLINE	52
5.7	ENCRYPTING SECRET DATA AT REST	52
5.7.1	COMMANDLINE	52
5.8	MULTI CONTAINER PODS.....	53
5.8.1	COMMANDLINE	54
5.9	MULTI CONTAINER PODS DESIGN PATTERNS.....	54
5.10	INIT CONTAINERS	54
5.10.1	COMMANDLINE	54
5.11	SELF HEALING APPLICATIONS	54
6.	CLUSTER MAINTENANCE.....	54
6.1	CLUSTER MAINTENANCE – SECTIONS INTRODUCTION	54
6.2	OPERATING SYSTEM UPGRADES.....	54
6.2.1	COMMANDLINE	56
6.3	KUBERNETES SOFTWARE VERSIONS	56
6.4	CLUSTER UPGRADE INTRODUCTION.....	58
6.4.1	COMMANDLINE	65
6.5	DEMO CLUSTER UPGRADE	65
6.5.1	COMMANDLINE	65
6.6	BACKUP AND RESTORE METHOD.....	66

6.6.1	COMMANDLINE	69
6.6.2	ETCD BACKUP AND RESTORE	69
6.7	MULTI NODE BACKUP (BACKUP AND RESTORE-2).....	70
6.7.1	COMMANDLINE	70
7.	SECURITY	71
7.1	SECURITY-SECTION INTRODUCTION	71
7.2	KUBERNETES SECURITY PRIMITIVES	72
7.3	AUTHENTICATION	72
7.4	ARTICLE ON SETTING UP BASIC AUTHENTICATION	72
7.5	TLS INTRODUCTION	72
7.6	TLS BASICS.....	72
7.7	TLS IN KUBERNETES	72
7.8	TLS IN KUBERNETES-CERTIFICATION CREATION	75
7.8.1	COMMANDLINE	80
7.9	VIEW CERTIFICATE DETAILS	80
7.10	CERTIFICATES API	80
7.10.1	COMMANDLINE	83
7.11	KUBE CONFIG	83
7.11.1	COMMANDLINE	86
7.12	API GROUPS	87
7.13	AUTHORIZATION.....	89
7.13.1	TYPES OF AUTHORIZATION	89
7.14	RBAC (ROLE BASED ACCESS CONTROLS).....	92
7.14.1	COMMANDLINE	94
7.15	CLUSTER ROLES	95
7.15.1	COMMANDLINE	97
7.16	SERVICE ACCOUNTS	98
7.16.1	COMMANDLINE	102
7.17	IMAGE SECURITY.....	103
7.17.1	COMMANDLINE	104
7.18	PRE-REQUISITE – SECURITY IN DOCKER.....	104
7.19	SECURITY CONTEXTS.....	106
7.19.1	COMMANDLINE	107
7.20	NETWORK POLICIES	107
7.21	DEVELOPING NETWORK POLICIES	111
7.21.1	COMMANDLINE	112
7.22	KUBECTX AND KUBENS	113

8. STORAGE.....	113
8.1 STORAGE – SECTION INTRODUCTION.....	113
8.2 INTRODUCTION TO DOCKER STORAGE.....	113
8.3 STORAGE IN DOCKER	113
8.3.1 TYPES OF PERSISTANT VOLUME	116
8.3.2 COMMANDLINE	116
8.4 VOLUME DRIVER PLUGIN IN DOCKER	117
8.5 CONTAINER STORAGE INTERFACE	118
8.6 VOLUMES	118
8.7 PERSISTANT VOLUMES.....	120
8.8 COMMANDLINE	123
8.9 PERSISTANT VOLUME CLAIMS	123
8.10 COMMAND LINE	125
8.11 USING PVC IN PODS	125
8.12 STORAGE CLASS	125
8.13 COMMANDLINE	130
9. NETWORKING	130
9.1 COMMANDLINE	130
10. KUBERNETES FUNDAMENTALS.....	133
11. KUBERNETES BASIC COMMAND.....	134
12. KUBENATES IMPERATIVES COMMANDS.....	135
13. KUBERTNETES CLUSTER SETUP ON UBUNTU.....	135
14. 137	
15. 139	
16. 141	

Table of Figures

FIGURE 1 - AUTHENTICATION TYPES.....	71
--------------------------------------	----

1.1 CORE CONCEPTS

- Cluster Architect
- API Primitives
- Services & Other Network Primitives

1.2 SCHEDULING

1.3 LOGGING MONITORING

1.4 APPLICATIONS LIFECYCLE MANAGEMENT

1.5 CLUSTER MAINTENANCE

1.6 SECURITY

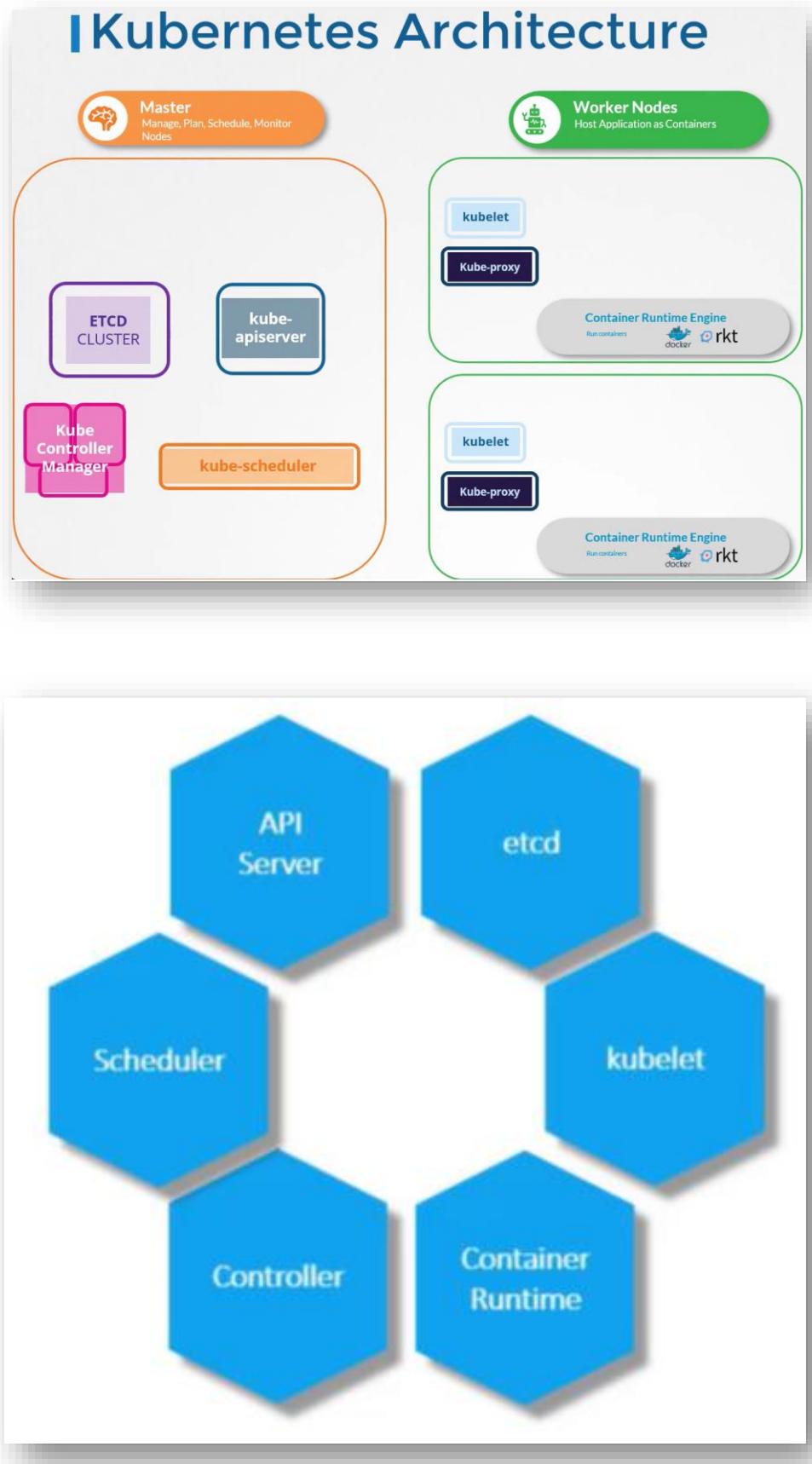
1.7 STORAGE

1.8 NETWORKING

1.9 INSTALLATION, CONFIGURATION & VALIDATION

1.10 TROUBLESHOOTING

2.1 CLUSTER ARCHITECTURE



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)

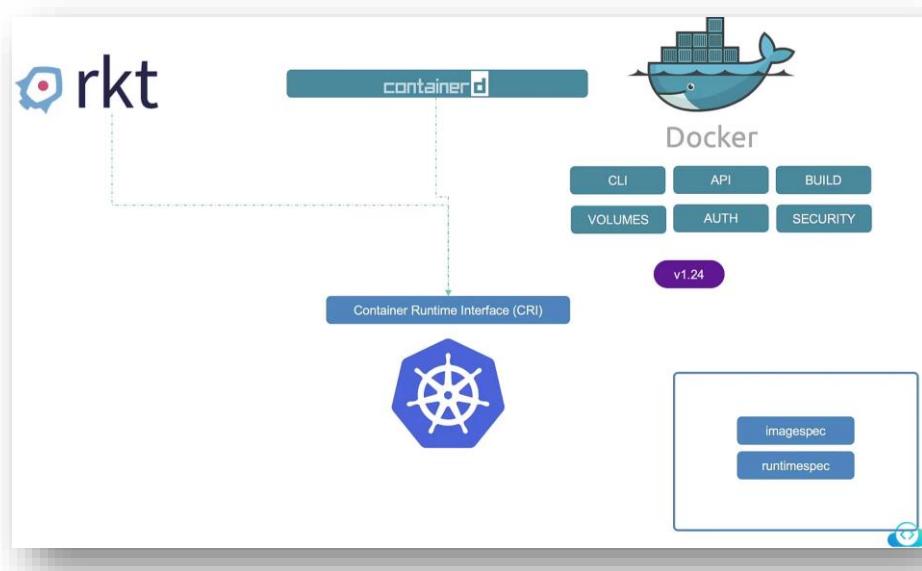
2.1.1.6 Container Runtime Engine (ContainerD, Rocket)

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	criclt	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

ETCD Versions

v0.1	Aug 2013
v0.5	Dec 2014
v2.0	Feb 2015
v3.1	Jan 2017
Nov 2018	CNCF Incubation

Install ETCD

1. Download Binaries

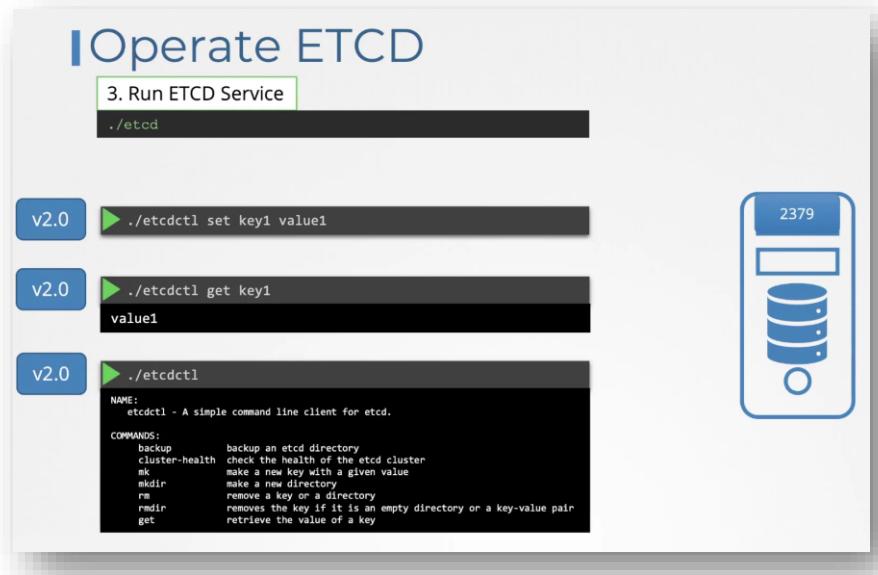
```
curl -L https://github.com/etcd-io/etcd/releases/download/v3.3.11/etcd-v3.3.11-linux-amd64.tar.gz -o etcd-v3.3.11-linux-amd64.tar.gz
```

2. Extract

```
tar xzvf etcd-v3.3.11-linux-amd64.tar.gz
```

3. Run ETCD Service

```
./etcd
```

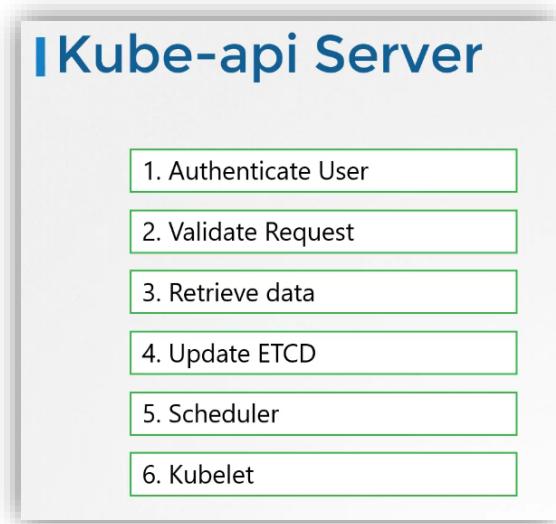


2.3.1 ETCD-COMMANDS

- 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

| View api-server - kubeadm

▶ kubectl get pods -n kube-system						
NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	
kube-system	coredns-78fcdf6894-hwrq9	1/1	Running	0	16m	
kube-system	coredns-78fcdf6894-rzhjr	1/1	Running	0	16m	
kube-system	etcd-master	1/1	Running	0	15m	
kube-system	kube-apiserver-master	1/1	Running	0	15m	
kube-system	kube-controller-manager-master	1/1	Running	0	15m	
kube-system	kube-proxy-lzt6f	1/1	Running	0	16m	
kube-system	kube-proxy-zm5qd	1/1	Running	0	16m	
kube-system	kube-scheduler-master	1/1	Running	0	15m	
kube-system	weave-net-29z42	2/2	Running	1	16m	
kube-system	weave-net-snmdl	2/2	Running	1	16m	-

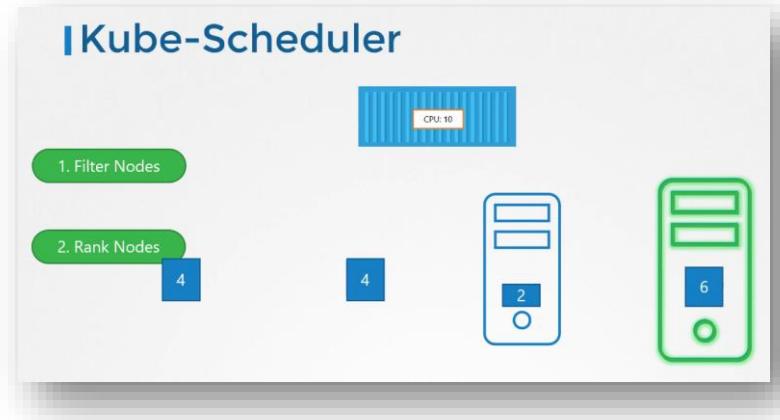
2.5 KUBECONTROL MANAGER

- Node Controller
- Replications Controller

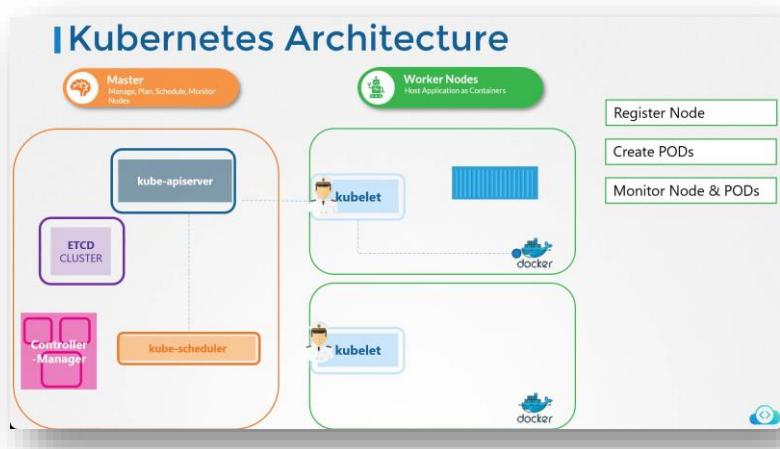
| View kube-controller-manager - kubeadm

▶ kubectl get pods -n kube-system						
NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	
kube-system	coredns-78fcdf6894-hwrq9	1/1	Running	0	16m	
kube-system	coredns-78fcdf6894-rzhjr	1/1	Running	0	16m	
kube-system	etcd-master	1/1	Running	0	15m	
kube-system	kube-apiserver-master	1/1	Running	0	15m	
kube-system	kube-controller-manager-master	1/1	Running	0	15m	
kube-system	kube-proxy-lzt6f	1/1	Running	0	16m	
kube-system	kube-proxy-zm5qd	1/1	Running	0	16m	
kube-system	kube-scheduler-master	1/1	Running	0	15m	
kube-system	weave-net-29z42	2/2	Running	1	16m	
kube-system	weave-net-snmdl	2/2	Running	1	16m	-

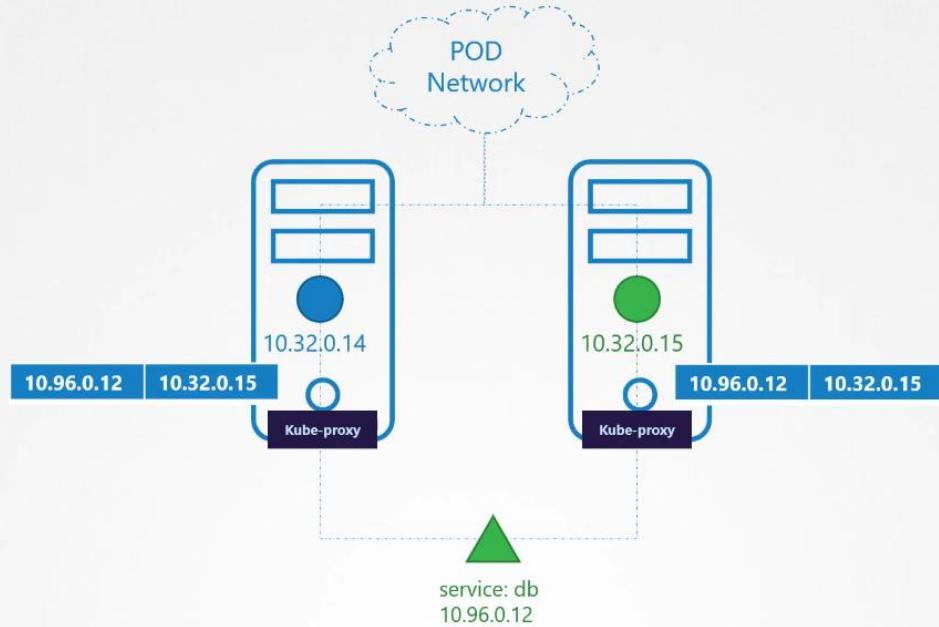
2.6 KUBE SCHEDULER



2.7 KUBELET



I Kube-proxy



I View kube-proxy - kubeadm

```
▶ kubectl get pods -n kube-system
  NAME           READY   STATUS    RESTARTS   AGE
  coredns-78fcdf6894-hwrq9   1/1     Running   0          16m
  coredns-78fcdf6894-rzhjr   1/1     Running   0          16m
  etcd-master      1/1     Running   0          15m
  kube-apiserver-master  1/1     Running   0          15m
  kube-controller-manager-master  1/1     Running   0          15m
  kube-proxy-lzt6f       1/1     Running   0          16m
  kube-proxy-zm5qd       1/1     Running   0          16m
  kube-scheduler-master  1/1     Running   0          15m
  weave-net-29z42        2/2     Running   1          16m
  weave-net-snmdl       2/2     Running   1          16m

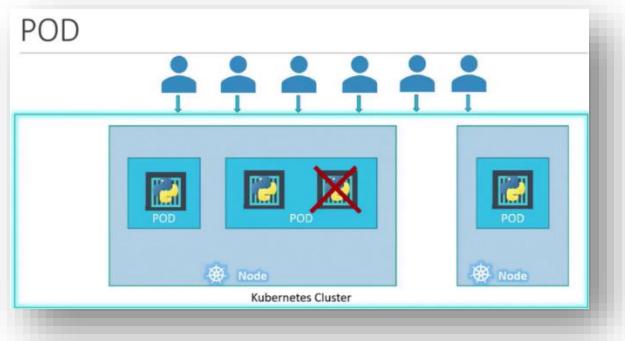
  ▶ kubectl get daemonset -n kube-system
  NAME            DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
  kube-proxy      2         2         2         2           2           beta.kubernetes.io/arch=amd64   1h
```

2.9 PODS

2.9.1 COMPOENETS

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

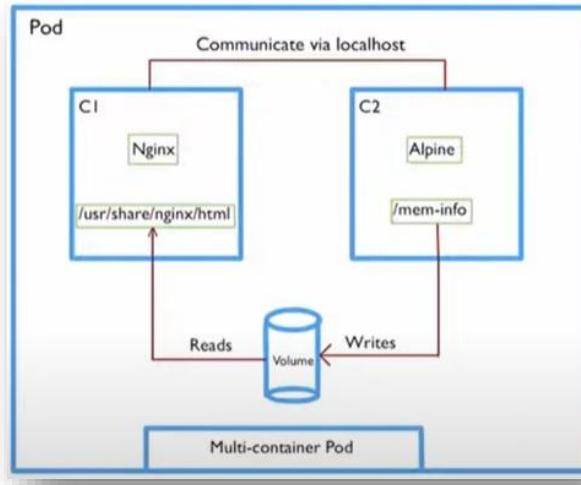
2.9.2 PODS STRUCTURE



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



```
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

```
pod-definition.yml
apiVersion:
kind:
metadata:
spec:
```

```
pod-definition.yml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
    type: front-end
spec:
  containers:
    - name: nginx-container
      image: nginx
```

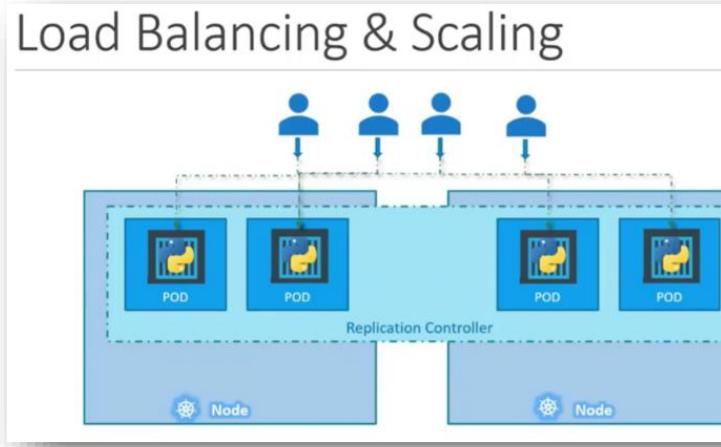
Kind	Version
POD	v1
Service	v1
ReplicaSet	apps/v1
Deployment	apps/v1

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

The screenshot shows a terminal window titled "Scale". It displays three command-line operations:

- > kubectl replace -f replicaset-definition.yml
- > kubectl scale --replicas=6 -f replicaset-definition.yml
- > kubectl scale --replicas=6 replicaset myapp-replicaset

Below these commands is a table with two columns: "TYPE" and "NAME". There are two entries in the table.

```
replicaset-definition.yml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-replicaset
  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
  replicas: 6
  selector:
    matchLabels:
      type: front-end
```

2.12.2 COMMANDLINE

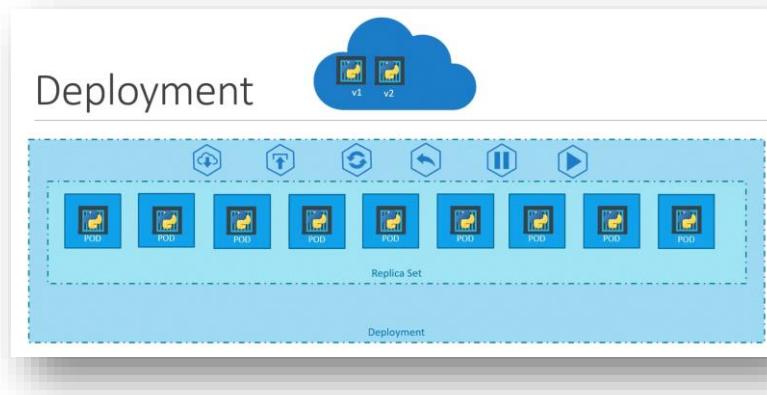
- kubectl api-resources | grep replicaset (To Check api Version)
- kubectl explain replicaset | grep VERSION (To Check replicaset version).
- kubectl create -f rc-definition.yml
- kubectl get replicaset
- kubectl describe replicaset myapp-replicaset
- kubectl replace -f rc-definition.yml
- kubectl edit replicaset 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 COMPONENTS

- Deploy
- Update
- Rollback

2.13.2



2.13.3

```
deployment-definition.yaml
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
  replicas: 3
  selector:
    matchLabels:
      type: front-end
```

```
> kubectl create -f deployment-definition.yaml
deployment "myapp-deployment" created

> kubectl get deployments
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
myapp-deployment   3         3         3           3           21s

> kubectl get replicaset
NAME      DESIRED   CURRENT   READY   AGE
myapp-deployment-6795844b58   3         3         3           2m
myapp-deployment-6795844b58-5rbjl   1/1       1/1       1           2m
myapp-deployment-6795844b58-h4w55   1/1       1/1       1           2m
myapp-deployment-6795844b58-lfjhv   1/1       1/1       1           2m

> kubectl get pods
NAME                           READY   STATUS   RESTARTS   AGE
myapp-deployment-6795844b58-5rbjl   1/1   Running   0          2m
myapp-deployment-6795844b58-h4w55   1/1   Running   0          2m
myapp-deployment-6795844b58-lfjhv   1/1   Running   0          2m
```

2.13.4

```
> kubectl get all
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/myapp-deployment   3         3         3           3           9h

NAME                  DESIRED   CURRENT   READY   AGE
rs/myapp-deployment-6795844b58   3         3         3           9h

NAME                           READY   STATUS   RESTARTS   AGE
po/myapp-deployment-6795844b58-5rbjl   1/1   Running   0          9h
po/myapp-deployment-6795844b58-h4w55   1/1   Running   0          9h
po/myapp-deployment-6795844b58-lfjhv   1/1   Running   0          9h
```

2.13.5 COMMAND LINE

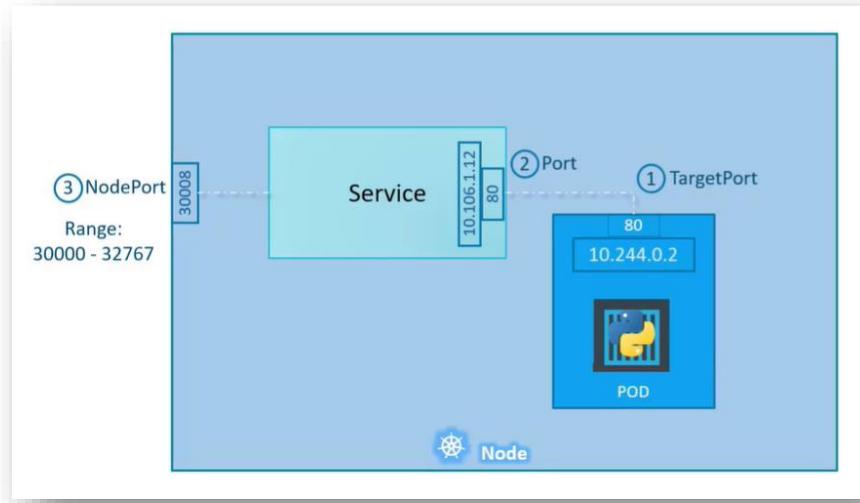
- `kubectl create deployment my-dep --image=nginx`
- `kubectl create -f deployment-denition.yaml`
- `kubectl get deployment`
- `kubectl describe deployment my-dep`
- `kubectl get replicaset`
- `kubectl get pods`

- `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-definition.yaml`
- `kubectl get deployment --watch`

2.14 SERVICES

2.14.1 TYPES OF SERVICES

- NodePort



```
service-definition.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
```

```

> kubectl create -f service-definition.yml
service "myapp-service" created

> kubectl get services
NAME         TYPE      CLUSTER-IP      EXTERNAL-IP   PORT(S)      AGE
kubernetes   ClusterIP  10.96.0.1     <none>        443/TCP     16d
myapp-service NodePort  10.106.127.123  <none>        80:30008/TCP  5m

> curl http://192.168.1.2:30008
<html>
<head>
<title>Welcome to nginx!</title>
<style>
body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
}
</style>
</head>
<body>

```

2.14.2 COMMANDLINE

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

2.14.3 CLUSTER IP

```

service-definition.yml

apiVersion: v1
kind: Service
metadata:
  name: back-end

spec:
  type: ClusterIP
  ports:
    - targetPort: 80
      port: 80

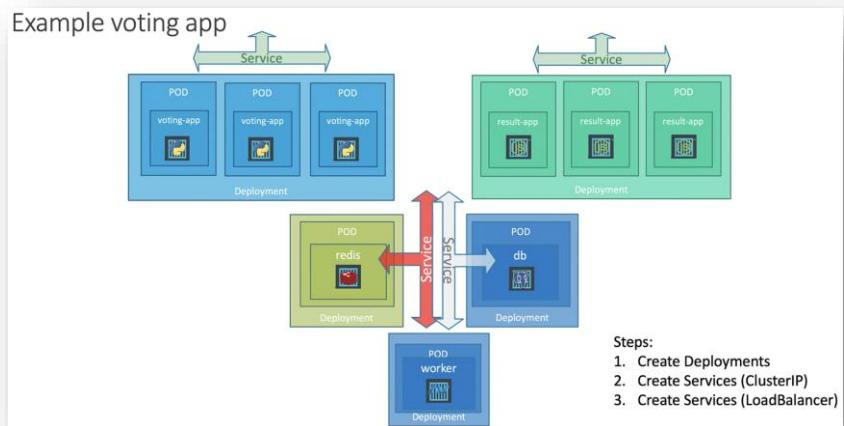
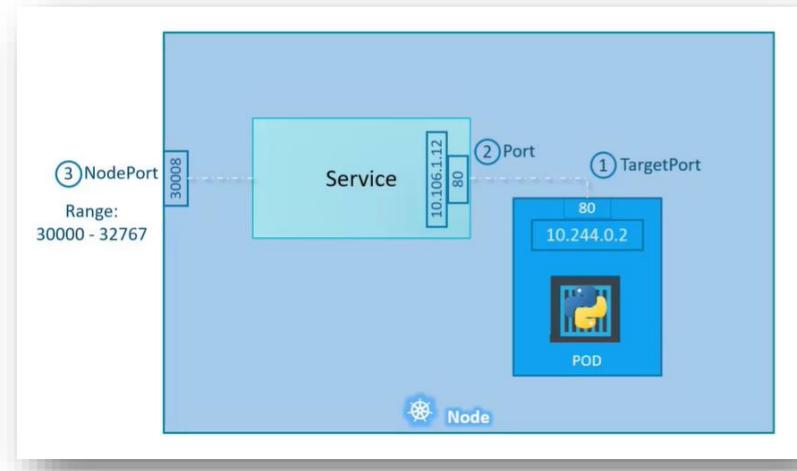
  selector:
    app: myapp
    type: back-end

```

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.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
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

Kubernetes

Imperative

```
> kubectl run --image=nginx nginx  
> kubectl create deployment --image=nginx nginx  
> kubectl expose deployment nginx --port 80  
> kubectl edit deployment nginx  
> kubectl scale deployment nginx --replicas=5  
> kubectl set image deployment nginx nginx=nginx:1.18  
> kubectl create -f nginx.yaml  
> kubectl replace -f nginx.yaml  
> kubectl delete -f nginx.yaml
```

Declarative

```
> kubectl apply -f nginx.yaml
```

Imperative Object Configuration Files

Create Objects

```
> kubectl create -f nginx.yaml
```

Update Objects

```
> kubectl edit deployment nginx  
> kubectl replace -f nginx.yaml  
> kubectl replace --force
```

```
nginx.yaml  
apiVersion: v1  
kind: Pod  
  
metadata:  
  name: myapp-pod  
  labels:  
    app: myapp  
    type: front-end-service  
spec:  
  containers:  
  - name: nginx-container  
    image: nginx:1.18
```

2.16.1 COMMANDLINE

3. SCHEDULING

3.1 MANUAL SCHEDULING

```
---  
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 LABELS AND SELECTORS

Labels

A diagram showing a single blue circular node. Below it, two labels are shown: 'app' with value 'App1' and 'function' with value 'Front-end'.

```
pod-definition.yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp
labels:
  app: App1
  function: Front-end

spec:
  containers:
  - name: simple-webapp
    image: simple-webapp
    ports:
      - containerPort: 8080
```

ReplicaSet

A diagram showing a purple rectangular application icon at the top. A dashed arrow points down to a row of three blue circular nodes, representing replicas.

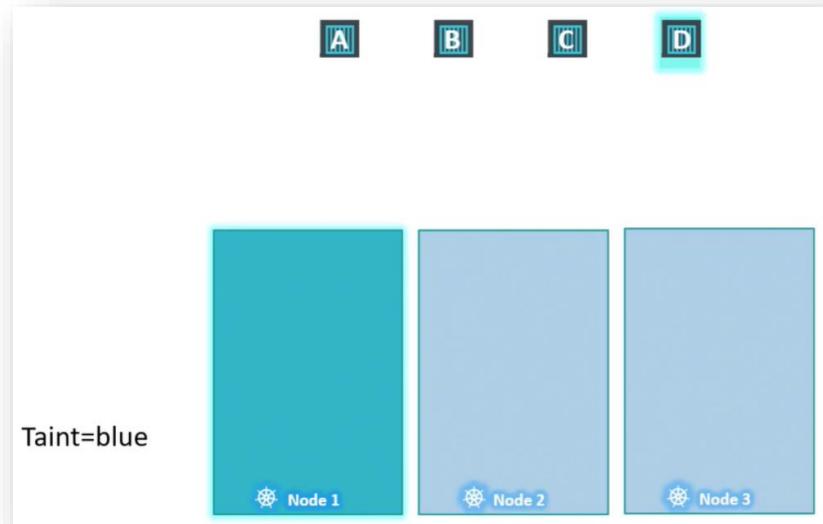
```
replicaset-definition.yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: simple-webapp
labels:
  app: App1
  function: Front-end
spec:
  replicas: 3
  selector:
    matchLabels:
      app: App1
  template:
    metadata:
      labels:
        app: App1
        function: Front-end
    spec:
      containers:
      - name: simple-webapp
        image: simple-webapp
```

3.2.1 COMMANDLINE

- kubectl label node node01 key=value (DefaultCommandad)
- 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 TRAINTS AND TOLERATIONS

- NoSchedule
- PreferNoSchedule
- NoExecute



⌚ Taints - Node

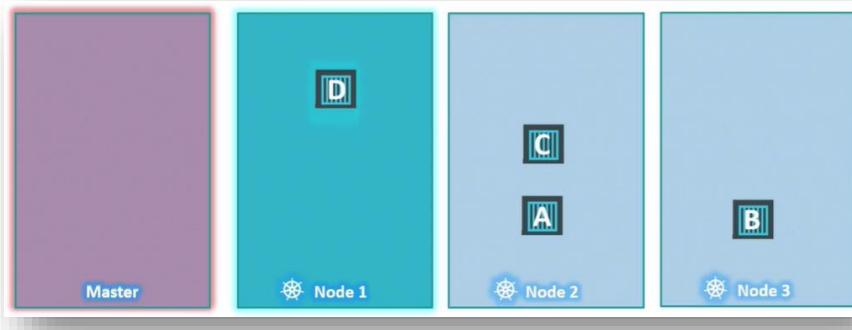
kubectl taint nodes node-name key=value:taint-effect

NoSchedule | PreferNoSchedule | NoExecute

What happens to PODs that do not tolerate this taint?

kubectl taint nodes node1 app=blue:NoSchedule

```
pod-definition.yml
apiVersion: v1
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`

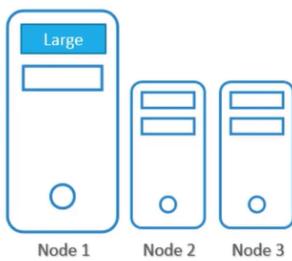
3.4 NODE SELECTOR

Node Selectors

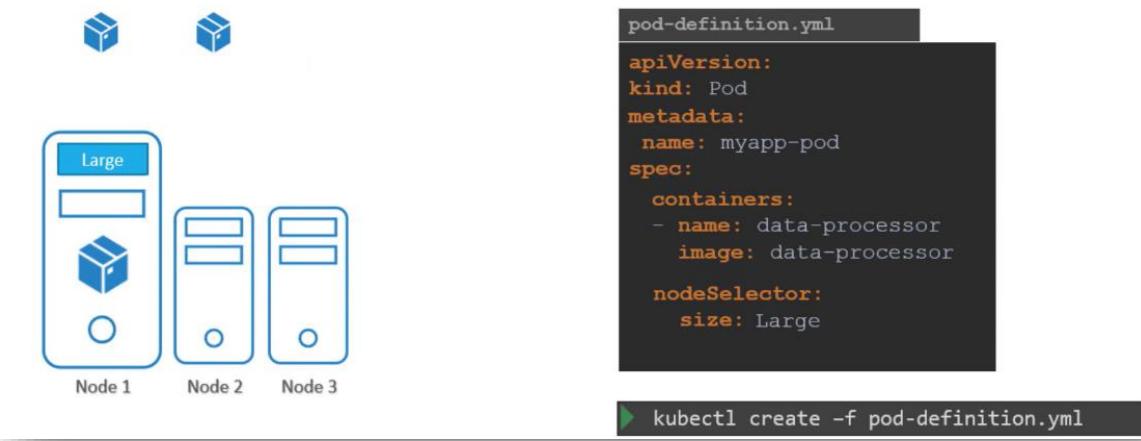
```
pod-definition.yml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
    - name: data-processor
      image: data-processor
  nodeSelector:
    size: Large
```

Label Nodes

```
▶ kubectl label nodes <node-name> <label-key>=<label-value>
▶ kubectl label nodes node-1 size=Large
```



Node Selector



3.4.1 COMMANDLINE

- kubectl label node node01 key=value (DefaultCommandad)
- 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

Node Affinity

```
pod-definition.yml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
    - name: data-processor
      image: data-processor
  nodeSelector:
    size: Large
```

```
pod-definition.yml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
    - name: data-processor
      image: data-processor
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: size
                operator: In
                values:
                  - Large
```

```

pod-definition.yml

apiVersion: v1
kind: Pod
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:

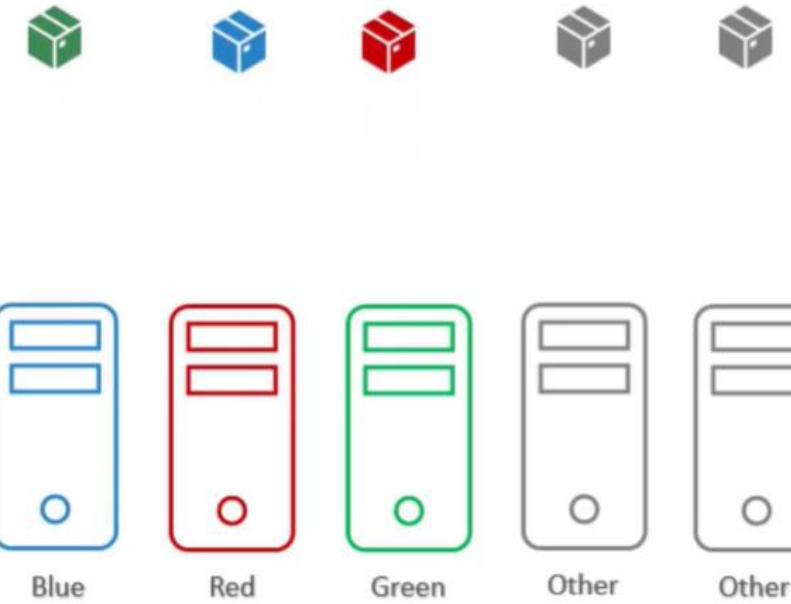
`requiredDuringSchedulingRequiredDuringExecution`

`preferredDuringSchedulingRequiredDuringExecution`

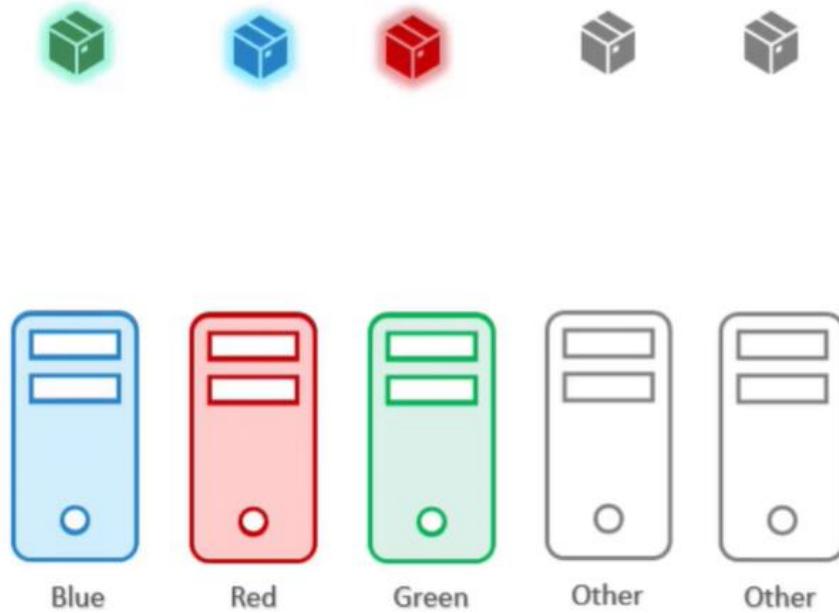
	DuringScheduling	DuringExecution
Type 1	Required	Ignored
Type 2	Preferred	Ignored
Type 3	Required	Required
Type 4	Preferred	Required



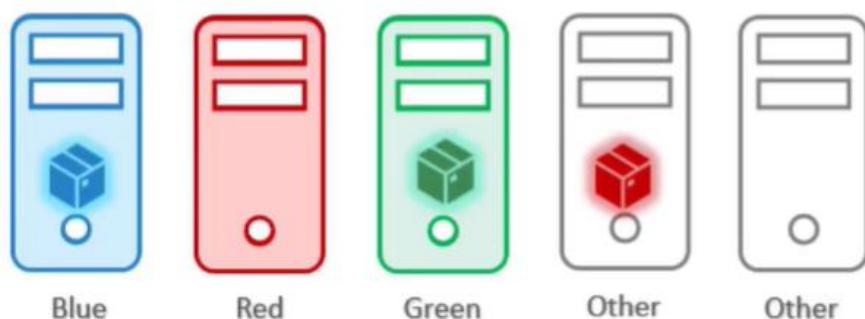
Taints and Tolerations



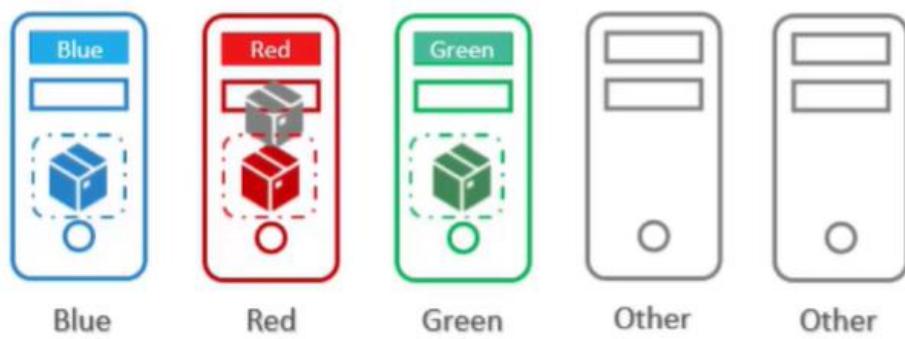
Taints and Tolerations



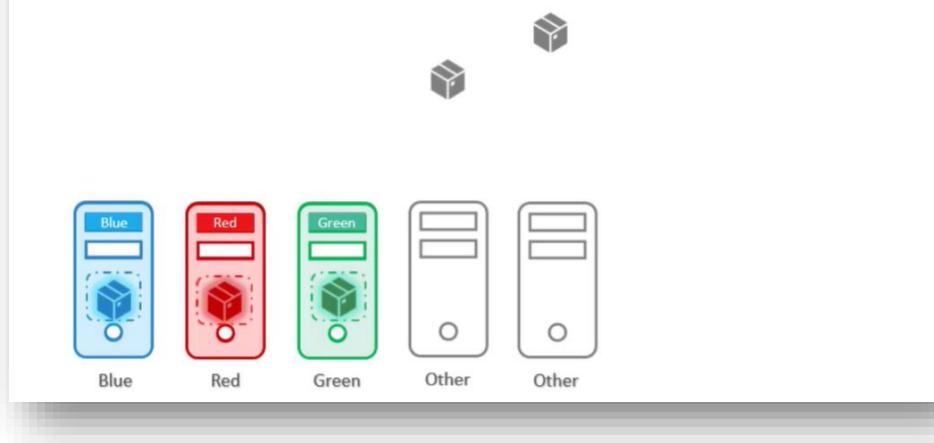
Taints and Tolerations



Node Affinity



Taints/Tolerations and Node Affinity



3.7 RESOURCE LIMIT

```
NAME     READY   STATUS    RESTARTS   AGE
Nginx   0/1     Pending   0          7m

Events:
Reason           Message
-----
FailedScheduling No nodes are available that match all of the following predicates:: Insufficient cpu (3).
```

pod-definition.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

```
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

```
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
```

I Behavior - CPU



I Behavior - Memory



I LimitRange

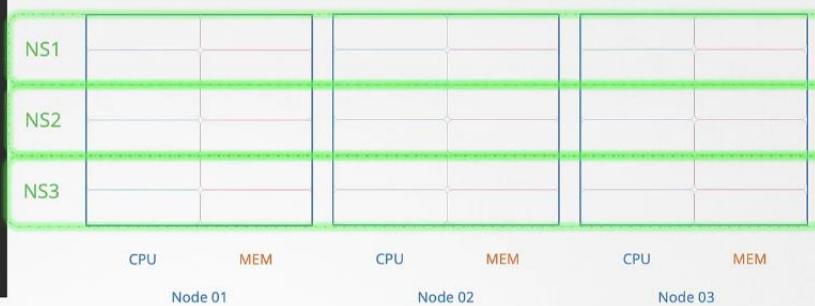
```
limit-range-cpu.yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-resource-constraint
spec:
  limits:
    - default:
        cpu: 500m
      defaultRequest:
        cpu: 500m
      max:
        cpu: "1"
      min:
        cpu: 100m
      type: Container
```

```
limit-range-memory.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
```

Resource Quotas

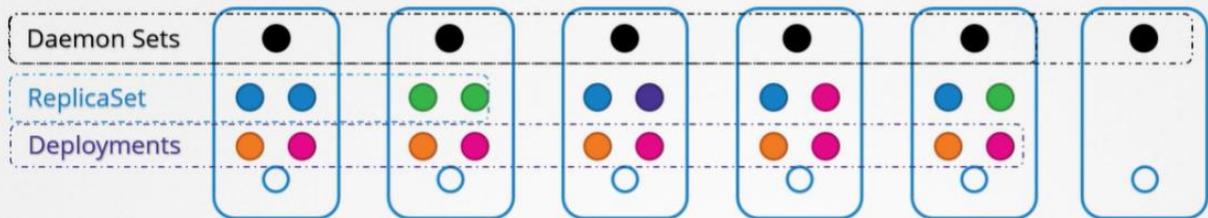
resource-quota.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: my-resource-quota
spec:
  hard:
    requests.cpu: 4
    requests.memory: 4Gi
    limits.cpu: 10
    limits.memory: 10Gi
```



3.8 DAEMON SET

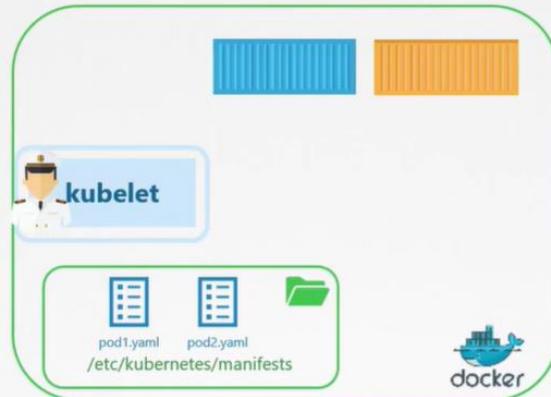
| Daemon Sets



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

I 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
```

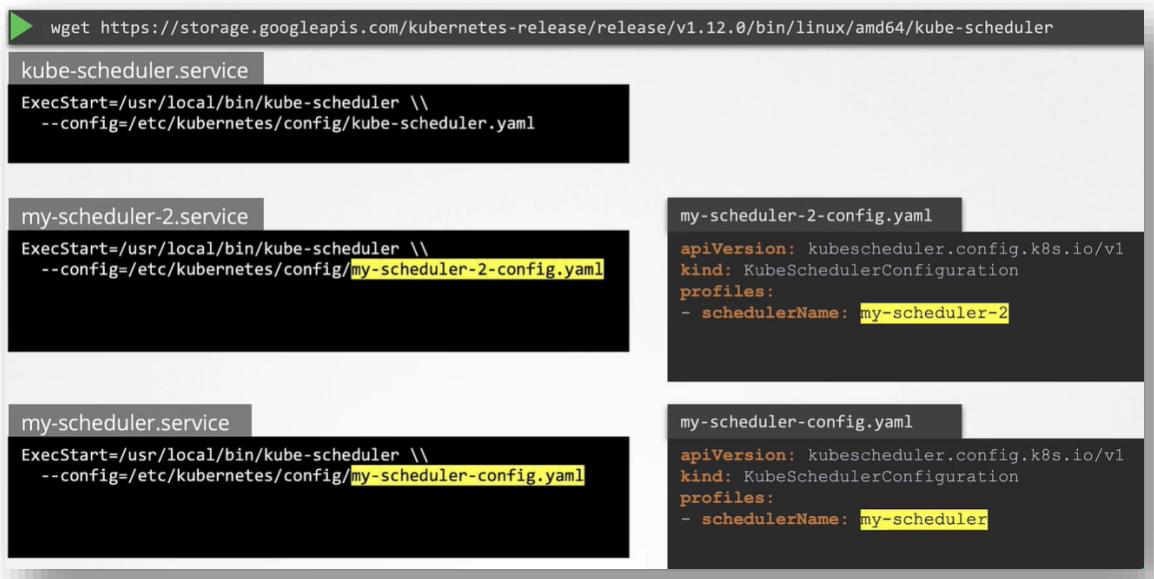
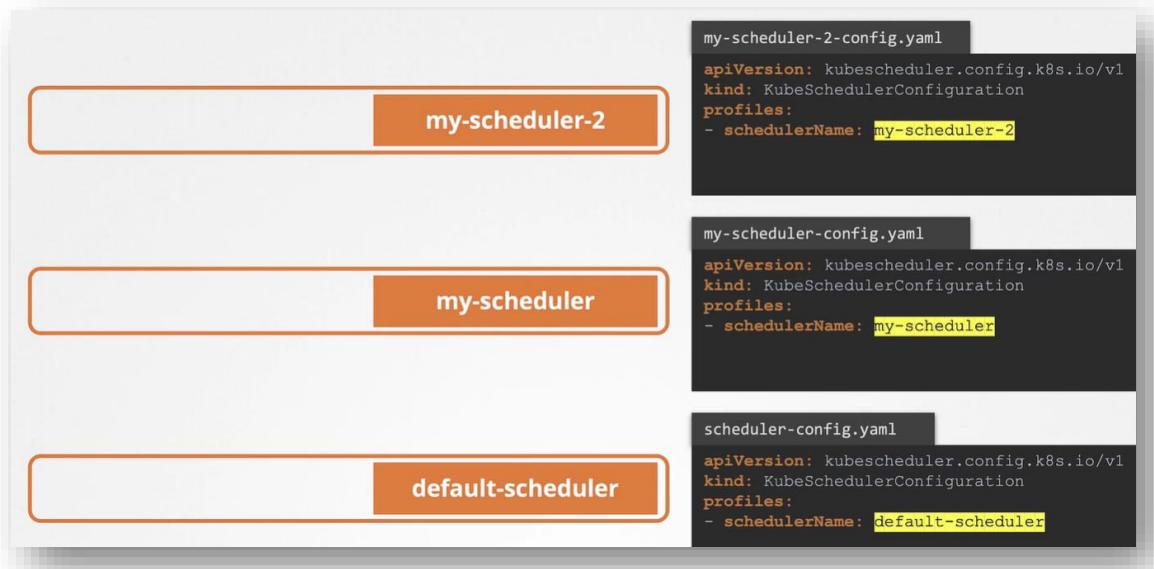
I Static PODs vs DaemonSets

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
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
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-dbgqv	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-dbgqv	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
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

Logs - Kubernetes

```
▶ kubectl create -f event-simulator.yaml  
  
▶ kubectl logs -f event-simulator-pod  
2018-10-06 15:57:15,937 - root - INFO - USER1 logged in  
2018-10-06 15:57:16,943 - root - INFO - USER2 logged out  
2018-10-06 15:57:17,944 - root - INFO - USER2 is viewing page2  
2018-10-06 15:57:18,951 - root - INFO - USER3 is viewing page3  
2018-10-06 15:57:19,954 - root - INFO - USER4 is viewing page1
```

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

event-simulator.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

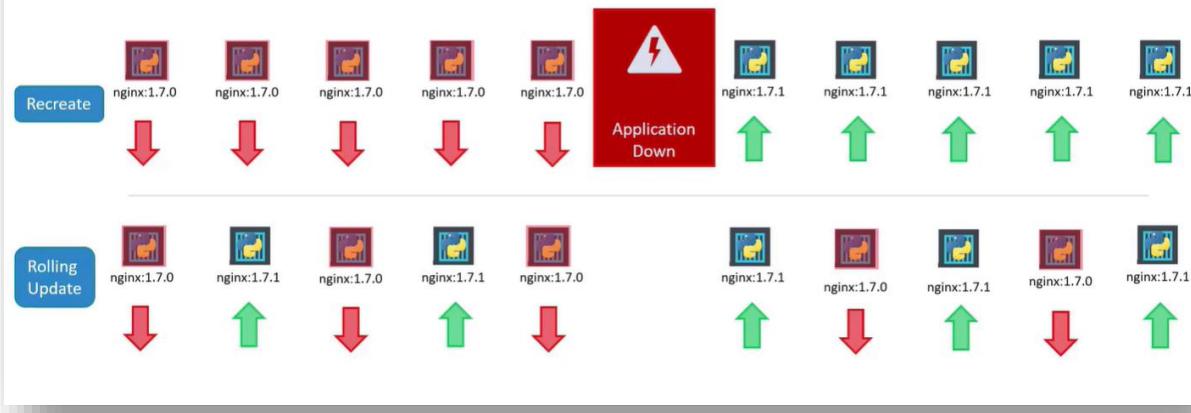
5.1 ROLLING UPDATES AND ROLLBACKS

Rollout and Versioning



| Revision 1 | nginx:1.7.0 |
|------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Revision 2 | nginx:1.7.1 |

Deployment Strategy



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
```

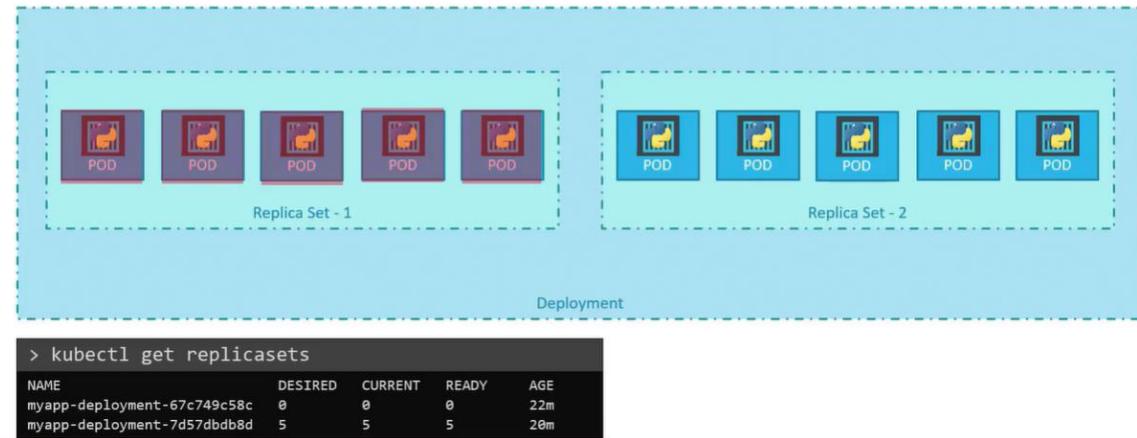
```
C:\Kubernetes>kubectl describe deployment myapp-deployment
Name:           myapp-deployment
Namespace:      default
CreationTimestamp: Sat, 03 Mar 2018 17:01:55 +0800
Labels:         app:myapp
Annotations:   deployment.kubernetes.io/revision:2
               kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"Deployment","metad...
File\Google...
               kubernetes.io/change-cause:kubectl apply --filename=d:\Vumshad Files\Google Drive\Udemy\Kubernetes\...
Selector:      type:front-end
Replicas:      5 desired | 5 updated | 5 total | 5 available | 0 unavailable
StrategyType:  Recreate
MinReadySeconds: 0
Pod Template:
  Labels:  app:myapp
           type:front-end
  Containers:
    nginx-container:
      Image:  nginx:1.7.1
      Port:   <none>
      Environment:  <none>
      Mounts:  <none>
      Volumes: <none>
  Conditions:
    Type Status Reason
    ----
    Available True  MinimumReplicasAvailable
    Progressing True  NewReplicaSetAvailable
OldReplicaSets: <none>
NewReplicaSet:  myapp-deployment-54c7d0bcc (5/5 replicas created)
Events:
  Type Reason     Age From           Message
  ----
  Normal ScalingReplicaSet 1m  deployment-controller  Scaled up replica set myapp-deployment-679584b58 to 5
  Normal ScalingReplicaSet 1m  deployment-controller  Scaled down replica set myapp-deployment-679584b58 to 0
  Normal ScalingReplicaSet 56s deployment-controller  Scaled up replica set myapp-deployment-54c7d0bcc to 5
```

```
C:\Kubernetes>kubectl describe deployment myapp-deployment
Name:           myapp-deployment
Namespace:      default
CreationTimestamp: Sat, 03 Mar 2018 17:16:53 +0800
Labels:         app:myapp
Annotations:   deployment.kubernetes.io/revision:2
               kubernetes.io/change-cause:kubectl apply --filename=d:\Vumshad Files\Google Drive\Udemy\Kubernetes\...
File\Google...
               kubernetes.io/change-cause:kubectl apply --filename=d:\Vumshad Files\Google Drive\Udemy\Kubernetes\...
Selector:      type:front-end
Replicas:      5 desired | 5 updated | 6 total | 4 available | 2 unavailable
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app:myapp
           type:front-end
  Containers:
    nginx-container:
      Image:  nginx
      Port:   <none>
      Environment:  <none>
      Mounts:  <none>
      Volumes: <none>
  Conditions:
    Type Status Reason
    ----
    Available True  MinimumReplicasAvailable
    Progressing True  ReplicasUpdated
OldReplicaSets: myapp-deployment-67c749c58c (1/1 replicas created)
NewReplicaSet:  myapp-deployment-7d57dbdb8d (5/5 replicas created)
Events:
  Type Reason     Age From           Message
  ----
  Normal ScalingReplicaSet 1m  deployment-controller  Scaled up replica set myapp-deployment-67c749c58c to 5
  Normal ScalingReplicaSet 1s  deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 2
  Normal ScalingReplicaSet 1s  deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 4
  Normal ScalingReplicaSet 1s  deployment-controller  Scaled up replica set myapp-deployment-67c749c58c to 3
  Normal ScalingReplicaSet 0s  deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 3
  Normal ScalingReplicaSet 0s  deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 4
  Normal ScalingReplicaSet 0s  deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 2
  Normal ScalingReplicaSet 0s  deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 5
  Normal ScalingReplicaSet 0s  deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 1
```

Recreate

RollingUpdate

Upgrades



Rollback



Summarize Commands

Create

```
> kubectl create -f deployment-definition.yml
```

Get

```
> kubectl get deployments
```

Update

```
> kubectl apply -f deployment-definition.yml
```

Status

```
> kubectl set image deployment/myapp-deployment nginx=nginx:1.9.1
```

```
> 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

The screenshot shows a terminal window with three distinct sections. The top section contains a Dockerfile snippet with `FROM Ubuntu`, `CMD sleep 5`, and a green box labeled "Command at Startup: sleep 10". The middle section contains a Dockerfile snippet with `FROM Ubuntu`, `ENTRYPOINT ["sleep"]`, and a green box labeled "Command at Startup: sleep 10". The bottom section contains a Dockerfile snippet with `FROM Ubuntu`, `ENTRYPOINT ["sleep"]`, `CMD ["5"]`, and a green box labeled "Command at Startup: sleep 5". Each section also includes a terminal command and its output:

- Top section: `docker run ubuntu-sleeper sleep 10` (Output: sleep 10)
- Middle section: `docker run ubuntu-sleeper 10` (Output: sleep: missing operand
Try 'sleep --help' for more information.)
- Bottom section: `docker run ubuntu-sleeper` (Output: sleep: missing operand
Try 'sleep --help' for more information.)

5.3 COMMANDS AND ARGUMENTS IN KUBERNETES

The screenshot shows a terminal window with three distinct sections. The top section contains a Dockerfile snippet with `FROM Ubuntu`, `ENTRYPOINT ["sleep"]`, and `CMD ["5"]`. The middle section contains a terminal command and its output: `docker run --name ubuntu-sleeper \ --entrypoint sleep2.0 ubuntu-sleeper 10` (Output: sleep2.0 10). The right section contains a YAML configuration file named `pod-definition.yml` with the following content:

```
pod-definition.yml
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-sleeper-pod
spec:
  containers:
    - name: ubuntu-sleeper
      image: ubuntu-sleeper
      args: ["10"]
```

Below the configuration file is a terminal command: `kubectl create -f pod-definition.yml`.

```
FROM Ubuntu
```

```
ENTRYPOINT ["sleep"]
```

```
CMD ["5"]
```

```
pod-definition.yml
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-sleeper-pod
spec:
  containers:
    - name: ubuntu-sleeper
      image: ubuntu-sleeper
      command: ["sleep2.0"]
      args: ["10"]
```

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

```
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"]

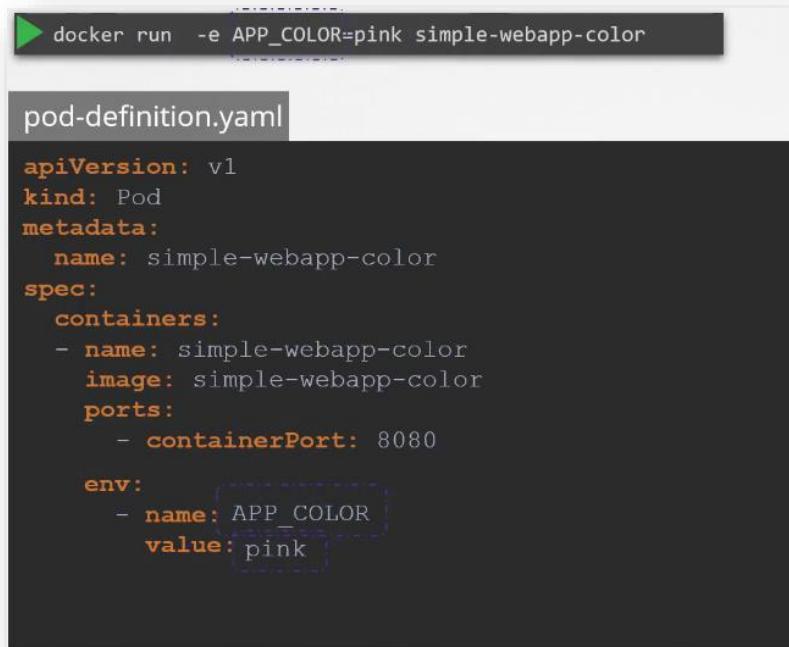
controlplane ~ ➔ cat webapp-color-3/webapp-color-pod-2.yaml
apiVersion: v1
kind: Pod
metadata:
  name: webapp-green
  labels:
    name: webapp-green
spec:
  containers:
    - name: simple-webapp
      image: kodekloud/webapp-color
      command: ["python", "app.py"]
      args: ["--color", "pink"]

```

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



```

▶ docker run -e APP_COLOR=pink simple-webapp-color

pod-definition.yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
spec:
  containers:
    - name: simple-webapp-color
      image: simple-webapp-color
      ports:
        - containerPort: 8080
      env:
        - name: APP_COLOR
          value: pink

```

config-map.yaml

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

ENV Value Types

```
env:
  - name: APP_COLOR
    value: pink
```

1 Plain Key Value

```
env:
  - name: APP_COLOR
    valueFrom:
      configMapKeyRef:
```

2 ConfigMap

```
env:
  - name: APP_COLOR
    valueFrom:
      secretKeyRef:
```

3 Secrets

5.5 CONFIGURE CONFIG MAP IN APPLICATIONS

Create ConfigMaps

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

Imperative

```
kubectl create configmap
<config-name> --from-literal=<key>=<value>
```

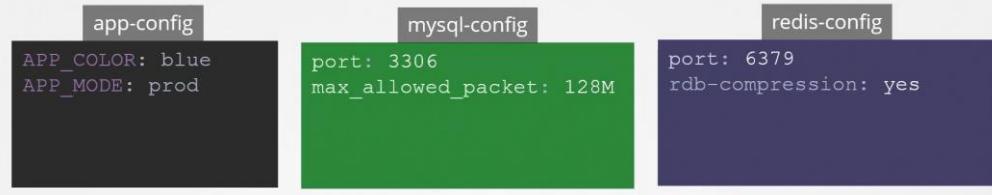
```
kubectl create configmap \
app-config --from-literal=APP_COLOR=blue \
--from-literal=APP_MODE=prod
```

1
Create ConfigMap

```
kubectl create configmap
<config-name> --from-file=<path-to-file>
```

```
kubectl create configmap \
app-config --from-file=app_config.properties
```

Create ConfigMaps



pod-definition.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

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

ENV

SINGLE ENV

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

```
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

| Web-MySQL Application

The screenshot shows two files side-by-side. On the left is `app.py`, which contains Python code for a Flask application. It connects to a MySQL database using root credentials and returns a template with a color value. On the right is `config-map.yaml`, a YAML file defining a ConfigMap named `app-config` with three key-value pairs: `DB_Host: mysql`, `DB_User: root`, and `DB_Password: paswrd`.

```
app.py
import os
from flask import Flask

app = Flask(__name__)

@app.route("/")
def main():

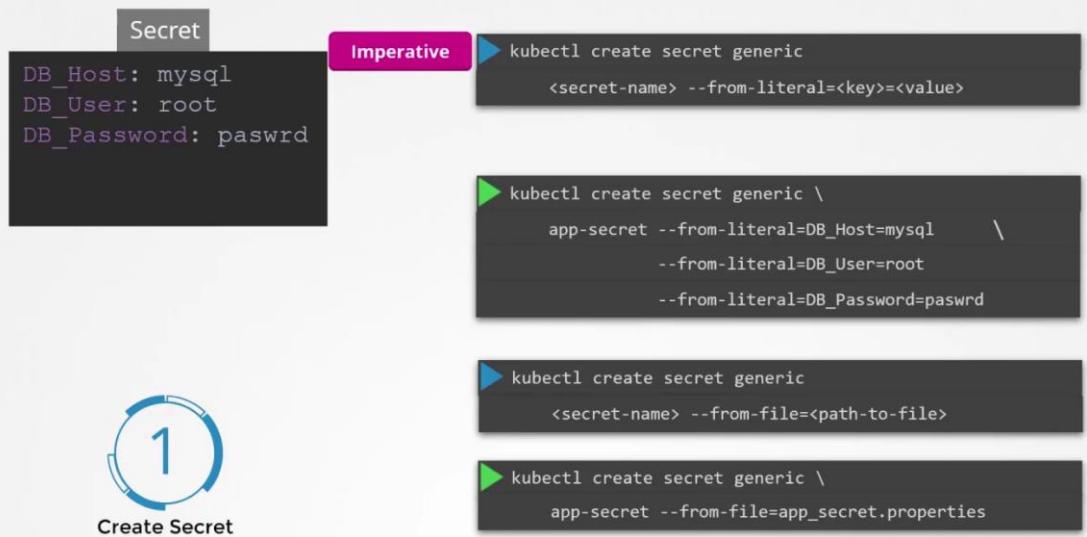
    mysql.connector.connect(host='mysql', database='mysql',
                           user='root', password='paswrd')

    return render_template('hello.html', color=fetchcolor())

if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```

```
config-map.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  DB_Host: mysql
  DB_User: root
  DB_Password: paswrd
```

| Create Secrets



I 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
```

I 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
```

I Secrets in Pods

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

ENV

SINGLE ENV

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

VOLUME

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

I 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

Note on Secrets

- ✖ Secrets are not Encrypted. Only encoded.
 - ✖ Do not check-in Secret objects to SCM along with code.
- ✖ Secrets are not encrypted in ETCD
 - ✓ Enable encryption at rest
- ✖ Anyone able to create pods/deployments in the same namespace can access the secrets
 - ✓ Configure least-privilege access to Secrets - RBAC
- ✓ Consider third-party secrets store providers
 - AWS Provider, Azure Provider, GCP Provider, Vault Provider

5.6.1 COMMANDLINE

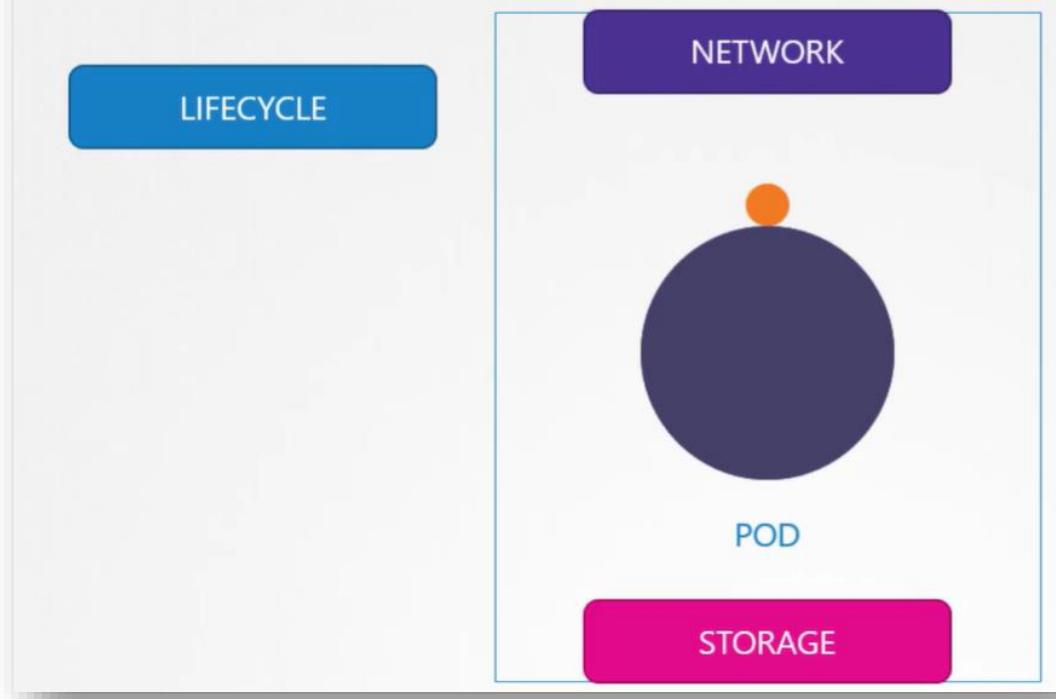
- kubectl get secrets
- kubectl describe secrets
- kubectl get secret app-secret -o yaml
- kubectl create secret generic secret_name --from-literal=key=value
- kubectl create secret generic my-secret --from-literal=DB_Host=mysql01 --from-literal=DB_User=root --from-literal=DB_Password=password123
-
- 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

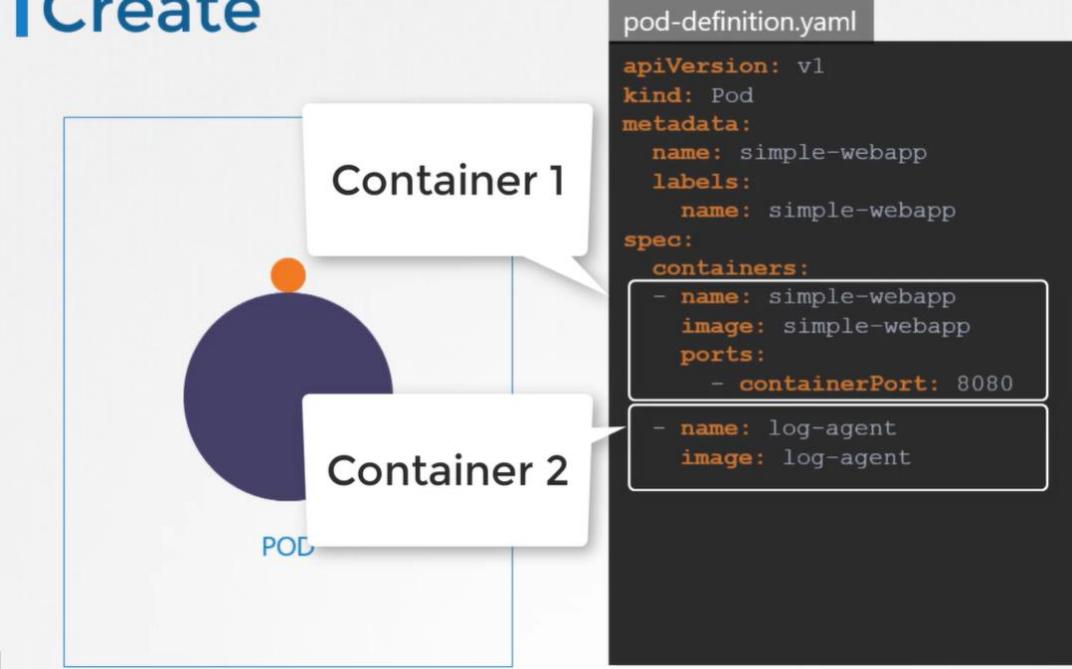
5.7.1 COMMANDLINE

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

| Multi-Container PODs



| Create



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

```
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

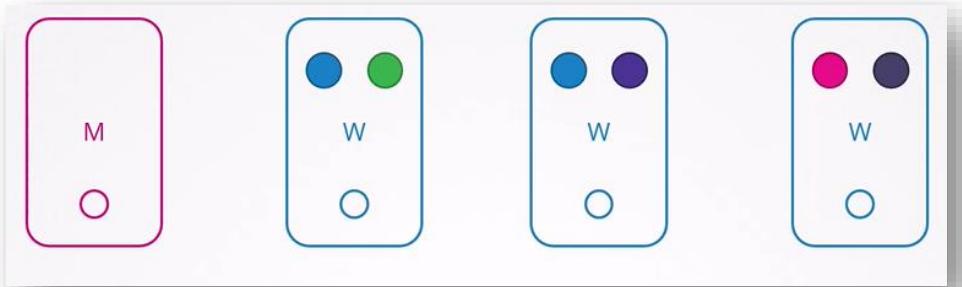
5.11 SELF HEALING APPLICATIONS

6. CLUSTER MAINTENANCE

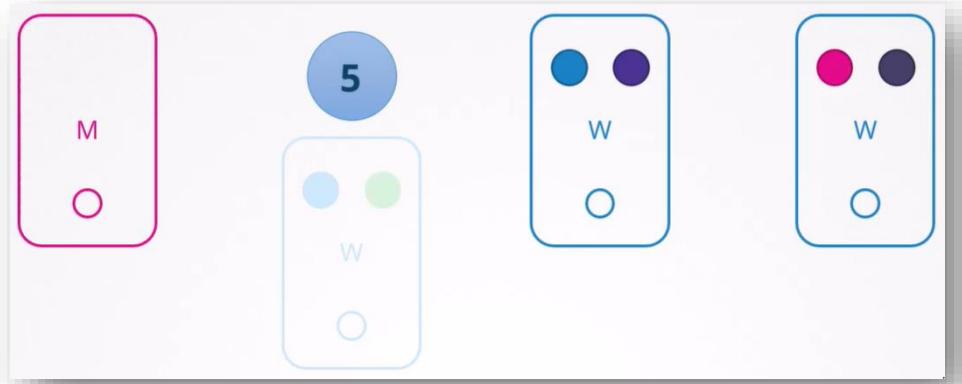
6.1 CLUSTER MAINTENANCE – SECTIONS INTRODUCTION

6.2 OPERATING SYSTEM UPGRADES

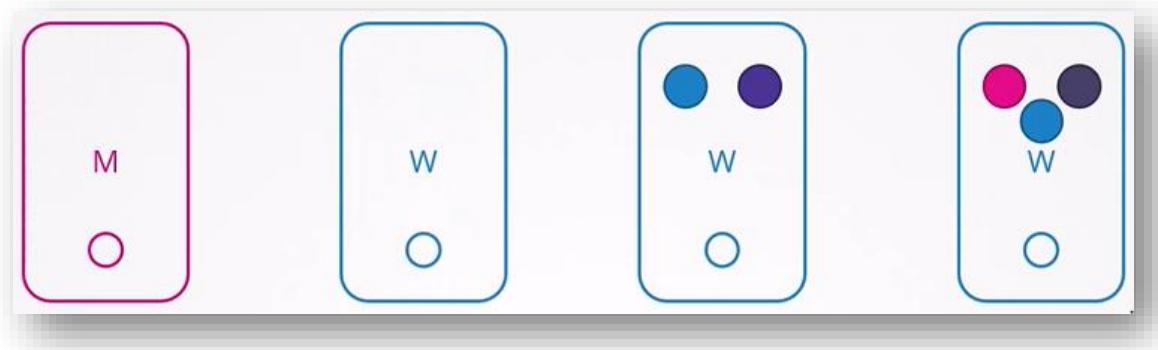
- Pod UP



- Pod Down.



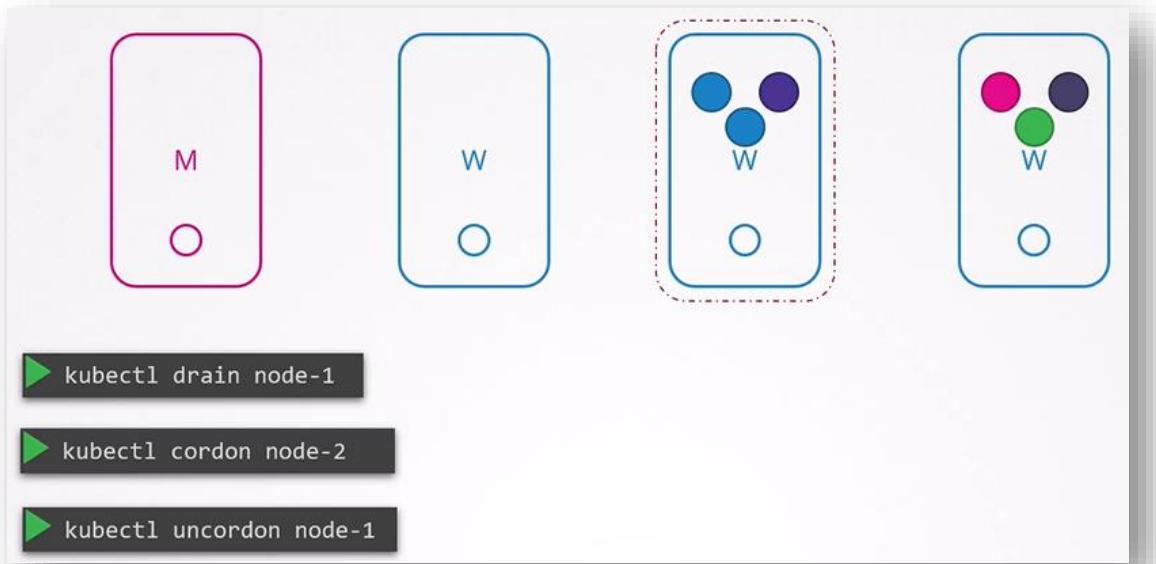
- Pod Recreate.



- Pod UP again



- Pod drain

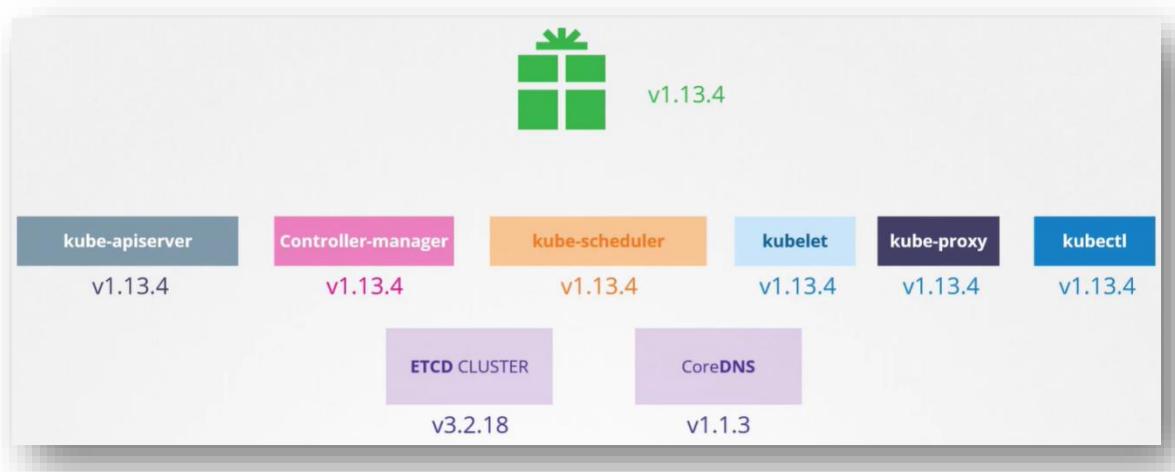
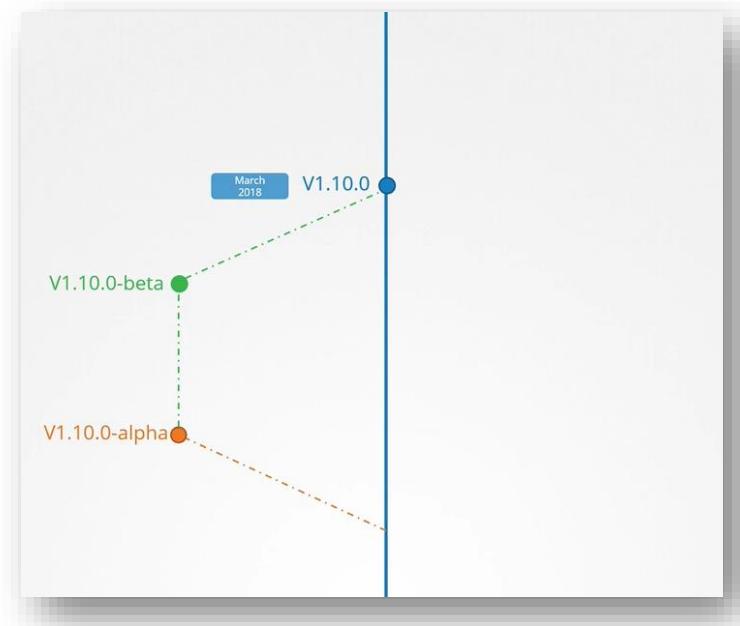
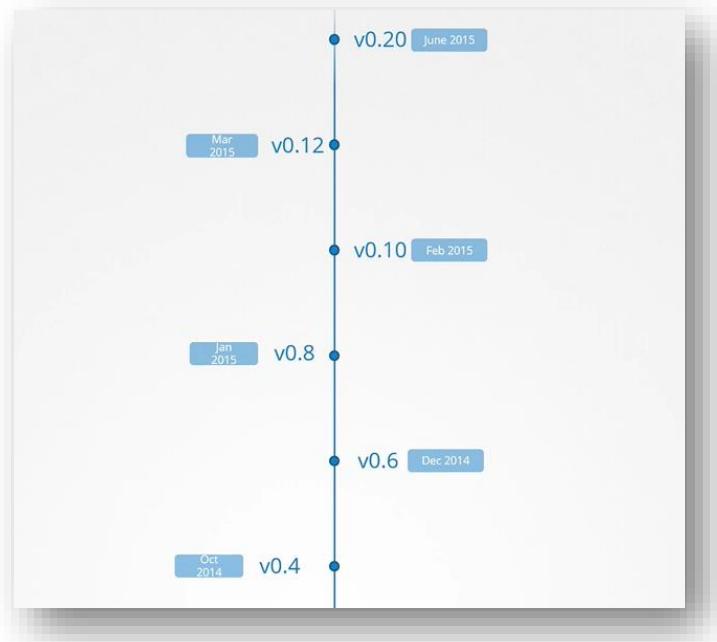


6.2.1 COMMANDLINE

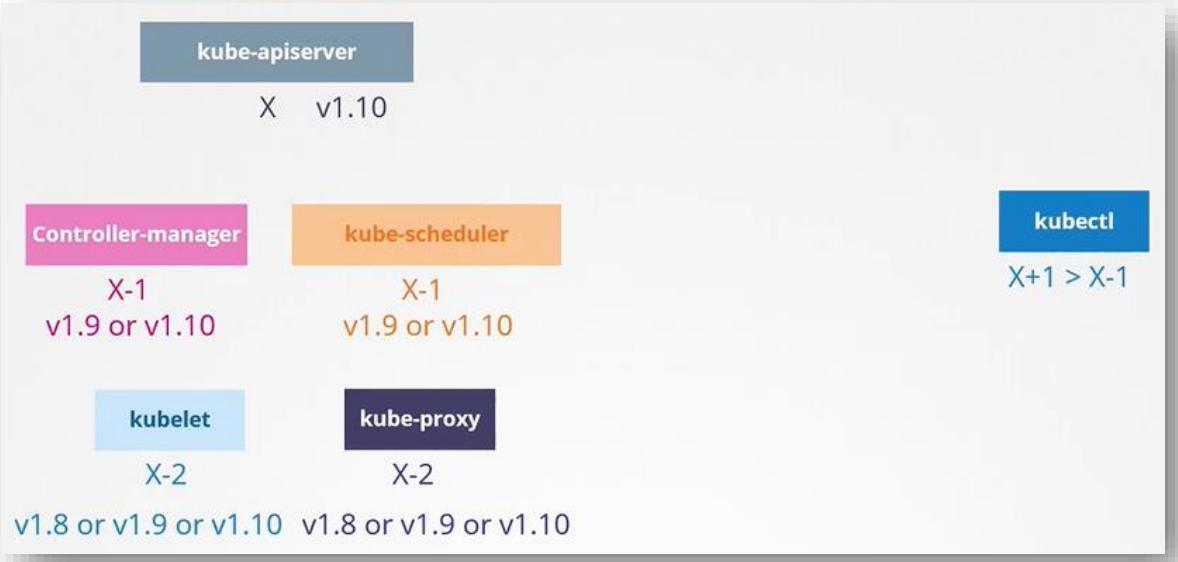
- `kubectl drain node01` (Move pod on other nodes).
- `Kubectl cordon node01` (Mark a node unscheduled)
- `Kubectl uncordon node01` (Mark a node Schedule)
- `kubectl drain node01 --ignore-daemonsets`

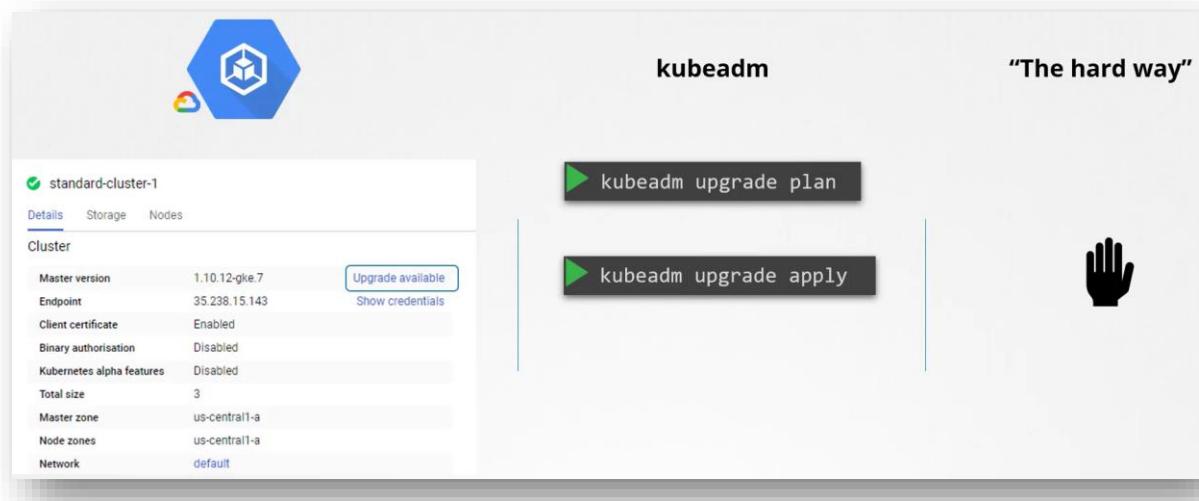
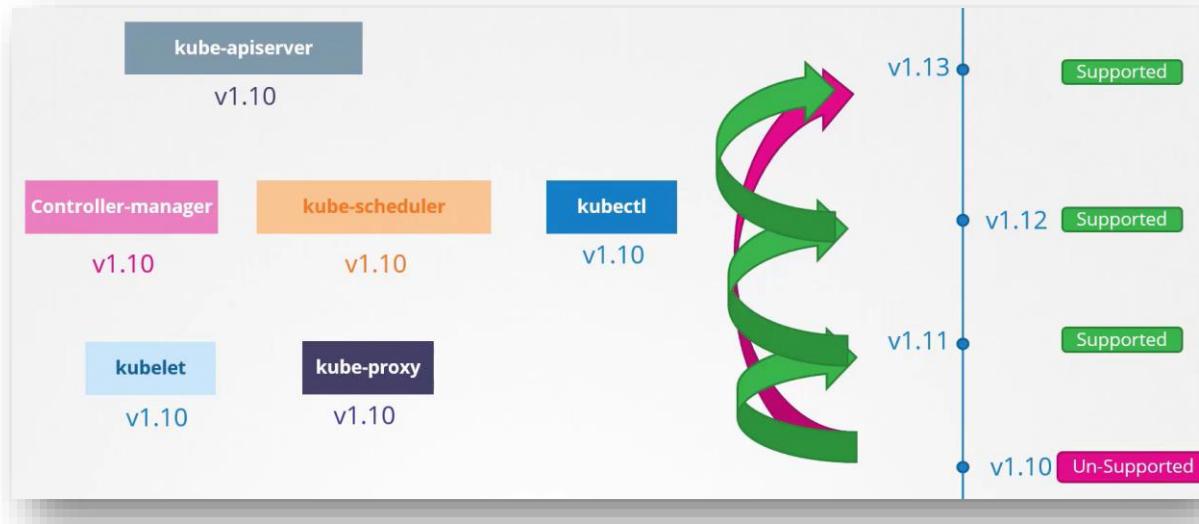
6.3 KUBERNETES SOFTWARE VERSIONS





6.4 CLUSTER UPGRADE INTRODUCTION

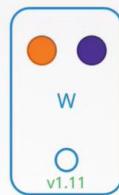
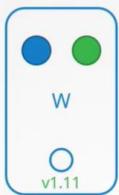




Strategy - 1



Strategy - 1



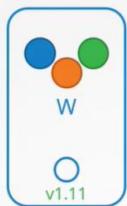
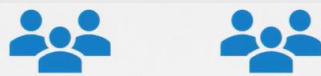
Strategy - 2



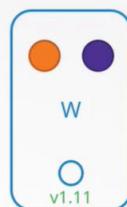
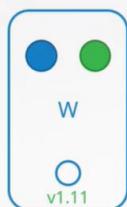
Strategy - 2



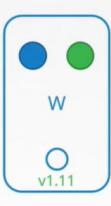
Strategy - 2



Strategy - 2



Strategy - 3



kubeadm - upgrade



```
▶ kubeadm upgrade plan
[preflight] Running pre-flight checks.
[upgrade] Making sure the cluster is healthy.
[upgrade/config] Making sure the configuration is correct.
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: v1.11.8
[upgrade/versions] kubeadm version: v1.11.3
[upgrade/versions] Latest stable version: v1.13.4
[upgrade/versions] Latest version in the v1.11 series: v1.11.8

Components that must be upgraded manually after you have
upgraded the control plane with 'kubeadm upgrade apply':
COMPONENT CURRENT AVAILABLE
Kubelet 3 x v1.11.3 v1.13.4

Upgrade to the latest stable version:
COMPONENT CURRENT AVAILABLE
API Server v1.11.8 v1.13.4
Controller Manager v1.11.8 v1.13.4
Scheduler v1.11.8 v1.13.4
Kube Proxy v1.11.8 v1.13.4
CoreDNS 1.1.3 1.1.3
Etcfd 3.2.18 N/A

You can now apply the upgrade by executing the following command:
```

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

kubeadm - upgrade



```
▶ apt-get upgrade -y kubeadm=1.12.0-00
▶ kubeadm upgrade apply v1.12.0
...
[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.12.0". Enjoy!
[upgrade/kubelet] Now that your control plane is upgraded, please proceed with
upgrading your kubelets if you haven't already done so.
```

kubeadm - upgrade

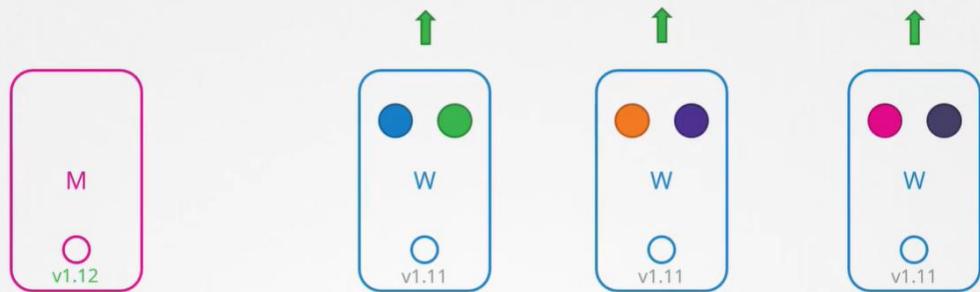


```
▶ kubectl get nodes
NAME STATUS ROLES AGE VERSION
master Ready master 1d v1.11.3
node-1 Ready <none> 1d v1.11.3
node-2 Ready <none> 1d v1.11.3

▶ apt-get upgrade -y kubelet=1.12.0-00
▶ systemctl restart kubelet

▶ kubectl get nodes
NAME STATUS ROLES AGE VERSION
master Ready master 1d v1.12.0
node-1 Ready <none> 1d v1.11.3
node-2 Ready <none> 1d v1.11.3
```

kubeadm - upgrade



kubeadm - upgrade



```
▶ kubectl drain node-1
```

kubeadm - upgrade



```
▶ 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



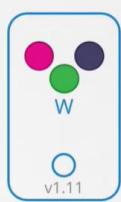
```
▶ kubectl drain node-1  
▶ kubectl uncordon node-1
```



```
▶ 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
```



```
▶ v1.11
```



```
▶ v1.11
```

kubeadm - upgrade



```
▶ kubectl drain node-1  
▶ kubectl uncordon node-1  
▶ kubectl drain node-2  
▶ kubectl uncordon node-2
```



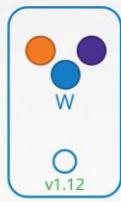
```
▶ v1.12  
▶ apt-get upgrade -y kubeadm=1.12.0-00  
▶ apt-get upgrade -y kubelet=1.12.0-00  
▶ kubeadm upgrade node config --kubelet-vers:  
▶ systemctl restart kubelet
```



kubeadm - upgrade



```
▶ kubectl drain node-1  
▶ kubectl uncordon node-1  
▶ kubectl drain node-2  
▶ kubectl uncordon node-2  
▶ kubectl drain node-3  
▶ kubectl uncordon node-3
```



```
v1.12
```



```
v1.12
```



```
v1.12
```

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

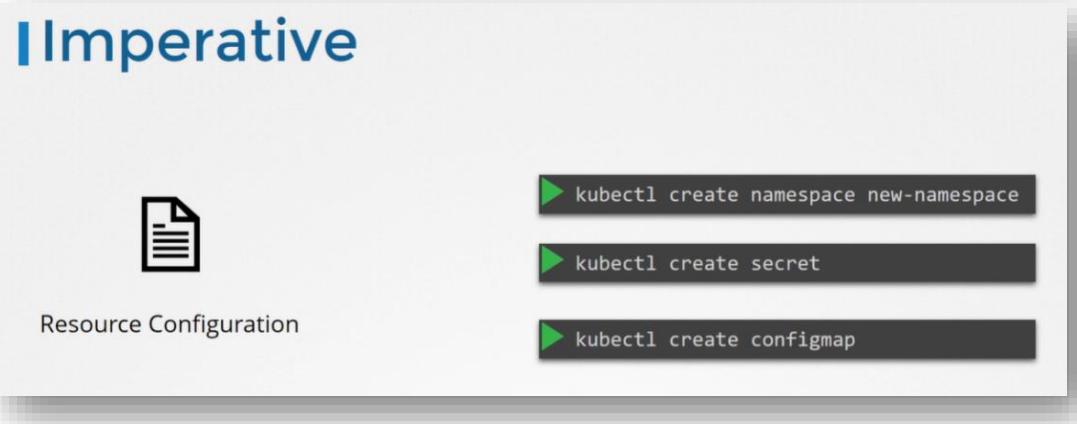
- cat /etc/*release*
- vim /etc/apt/sources.list.d/kubernetes.list
- deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
- apt-get update
- apt-cache madison kubeadm
- apt-get install kubeadm=1.30.0-1.1
- kubeadm version
- kubeadm upgrade plan v1.30.0
- 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



I Declarative



Resource Configuration

```
pod-definition.yml
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
```

I 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:2380 \
--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
```

I 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 |
+-----+-----+-----+-----+
```

I Restore - ETCD



ETCD Cluster

```
▶ ETCCTL_API=3 etcdctl \
snapshot restore snapshot.db \
--data-dir /var/lib/etcd-from-backup
I | mvcc: restore compact to 475629
```

```
▶ systemctl daemon-reload
▶ service etcd restart
Service etcd restarted
```

```
▶ ETCCTL_API=3 etcdctl \
snapshot save snapshot.db
ls
snapshot.db
▶ service kube-apiserver stop
Service kube-apiserver stopped
etc.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}:2380
--listen-peer-urls https:// ${INTERNAL_IP}:2380
--listen-client-urls https:// ${INTERNAL_IP}:2381
--advertise-client-urls https:// ${INTERNAL_IP}:2381
--initial-cluster-token etcd-cluster-0 \\
--initial-cluster controller-0=https:// ${CONTROLLER_IP}:2380
--initial-cluster-state new \\
--data-dir=/var/lib/etcd-from-backup
```

I Restore - ETCD



ETCD Cluster

```
▶ ETCCTL_API=3 etcdctl \
snapshot restore snapshot.db \
--data-dir /var/lib/etcd-from-backup
I | mvcc: restore compact to 475629
```

```
▶ systemctl daemon-reload
▶ service etcd restart
Service etcd restarted
```

```
▶ ETCCTL_API=3 etcdctl \
snapshot save snapshot.db
ls
snapshot.db
▶ service kube-apiserver stop
Service kube-apiserver stopped
```

```
▶ service kube-apiserver start
Service kube-apiserver started
```

```
▶ ETCCTL_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
```

IBackup Candidates



Resource Configuration



ETCD Cluster

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/etc.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:

I Authentication

Who can access?

- Files – Username and Passwords
- Files – Username and Tokens
- Certificates
- External Authentication providers - LDAP
- Service Accounts

Figure 1 - Authentication types.

Step 2:

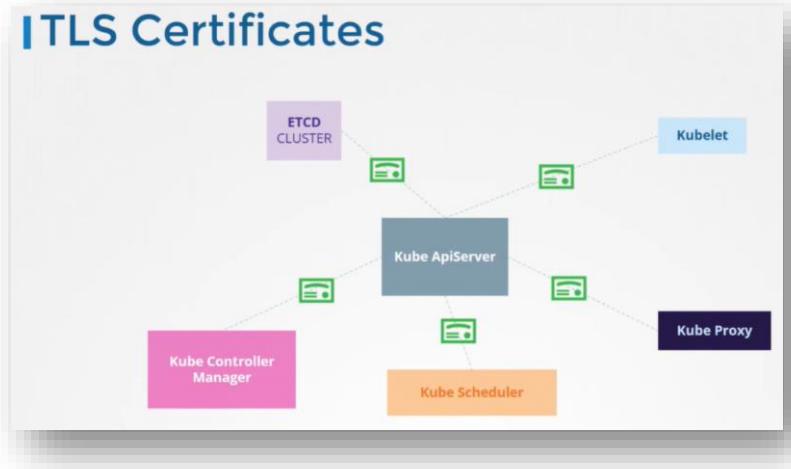
I Authorization

What can they do?

- RBAC Authorization
- ABAC Authorization
- Node Authorization
- Webhook Mode

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

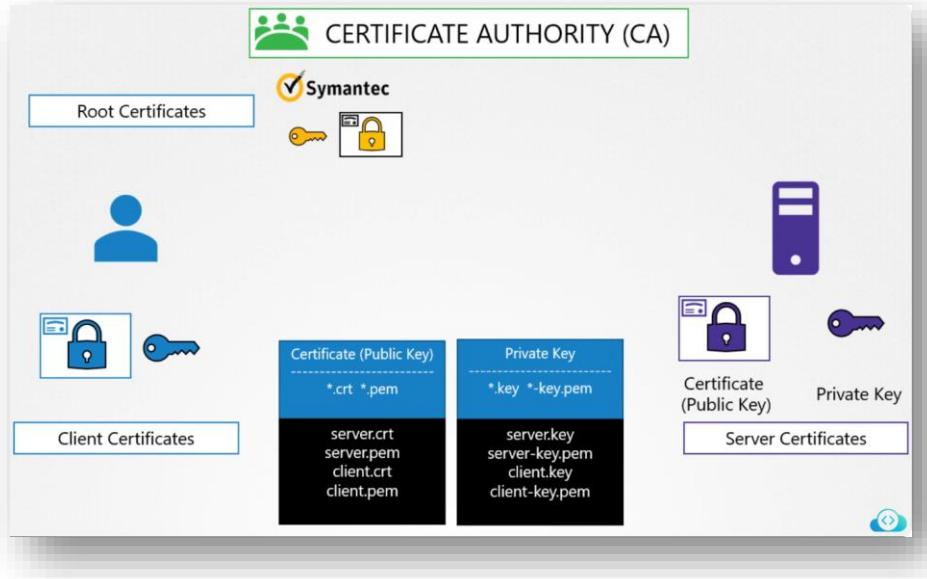
7.6 TLS BASICS

- SYMENTRIC
- ASYMENTRIC

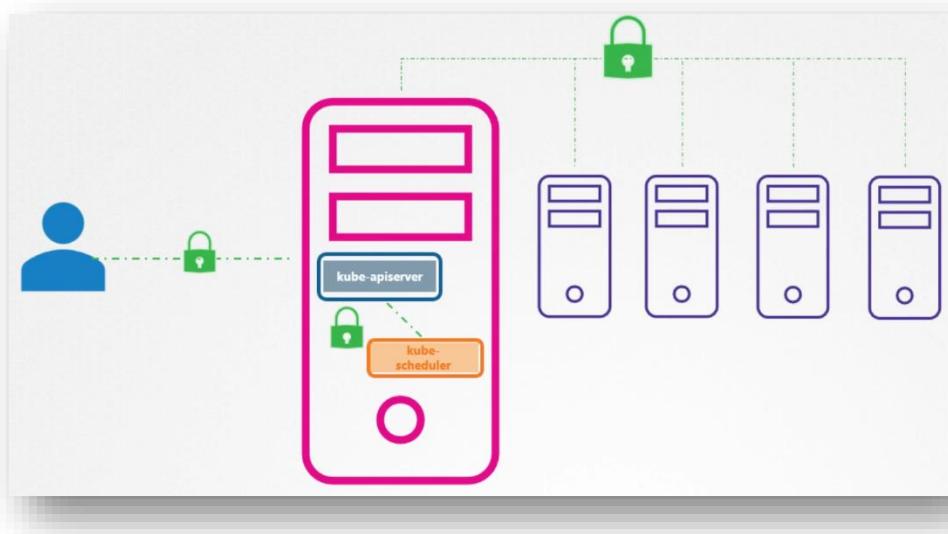
7.7 TLS IN KUBERNETES

- Server Certificate
- Root Certificates
- Client Certificates

Step 1.

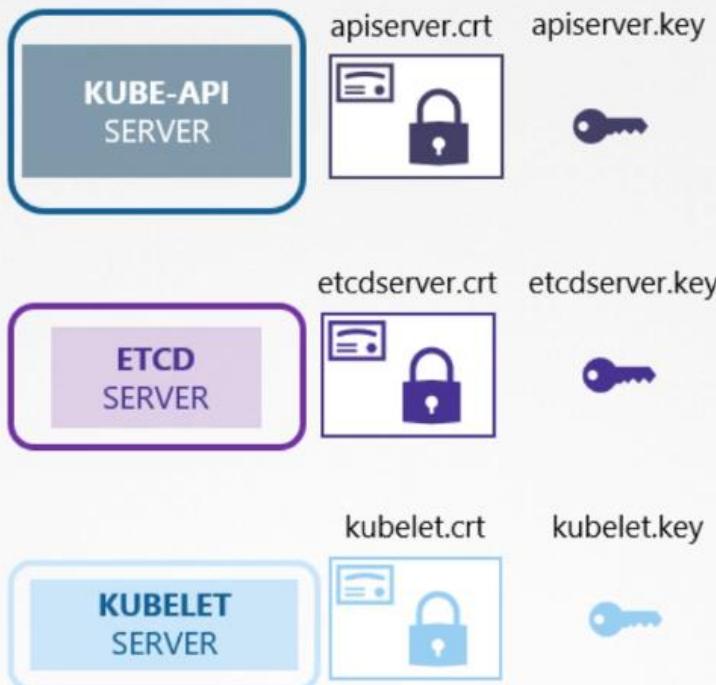


Step 4:

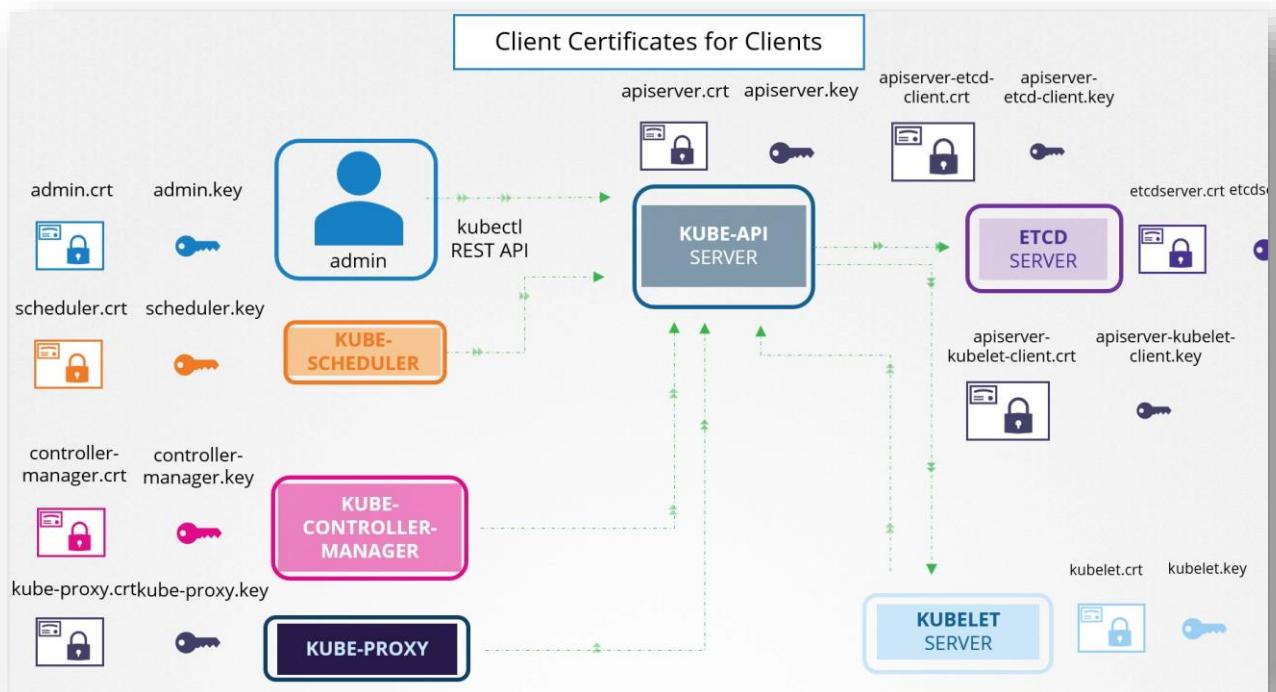


Step 5:

Server Certificates for Servers



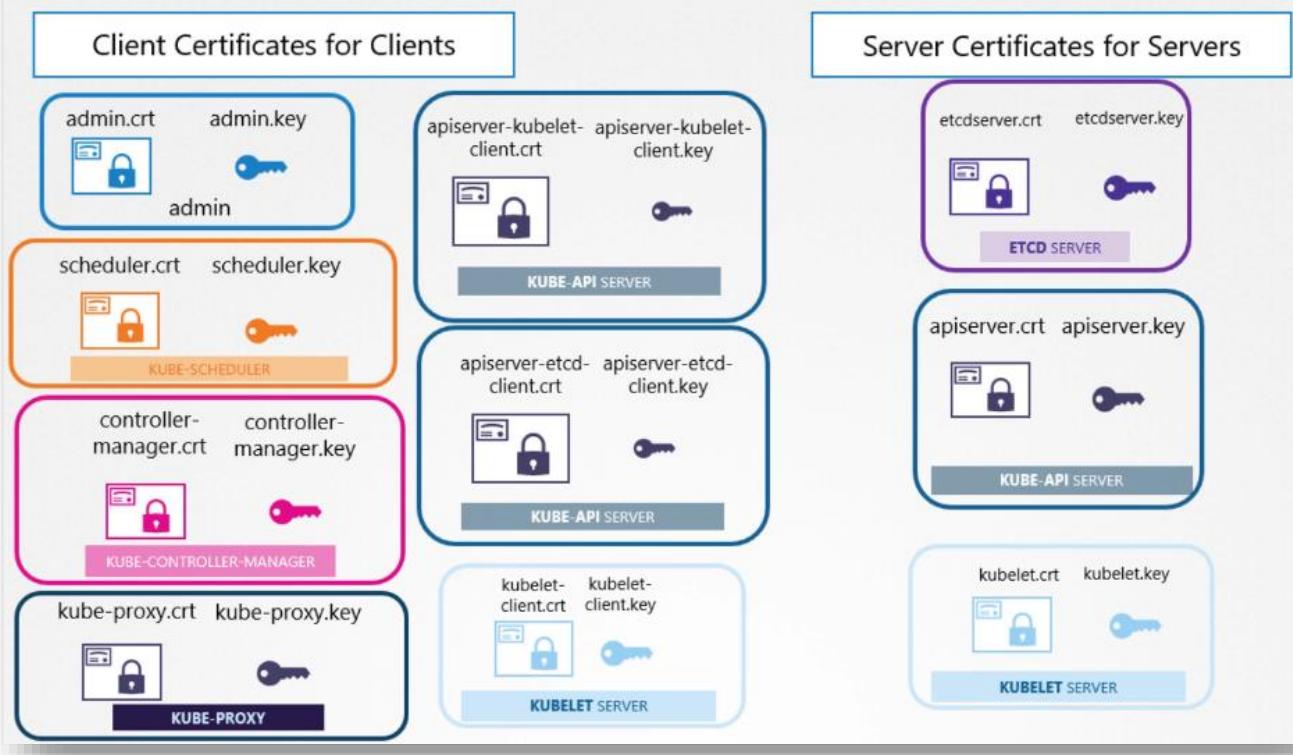
Step 2.



Step 3.



CERTIFICATE AUTHORITY (CA)



7.8 TLS IN KUBERNETES-CERTIFICATION CREATION

Step 1.



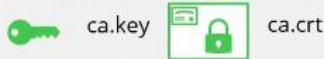
Step 2.



Step 3.



Step 4.



KUBE SCHEDULER

Generate Keys

scheduler.key



Certificate
Signing
Request

scheduler.csr

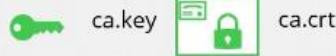


Sign
Certificates

scheduler.crt



Step 5.



KUBE CONTROLLER MANAGER

Generate Keys

controller-manager.key



Certificate
Signing
Request

controller-manager.csr



Sign
Certificates

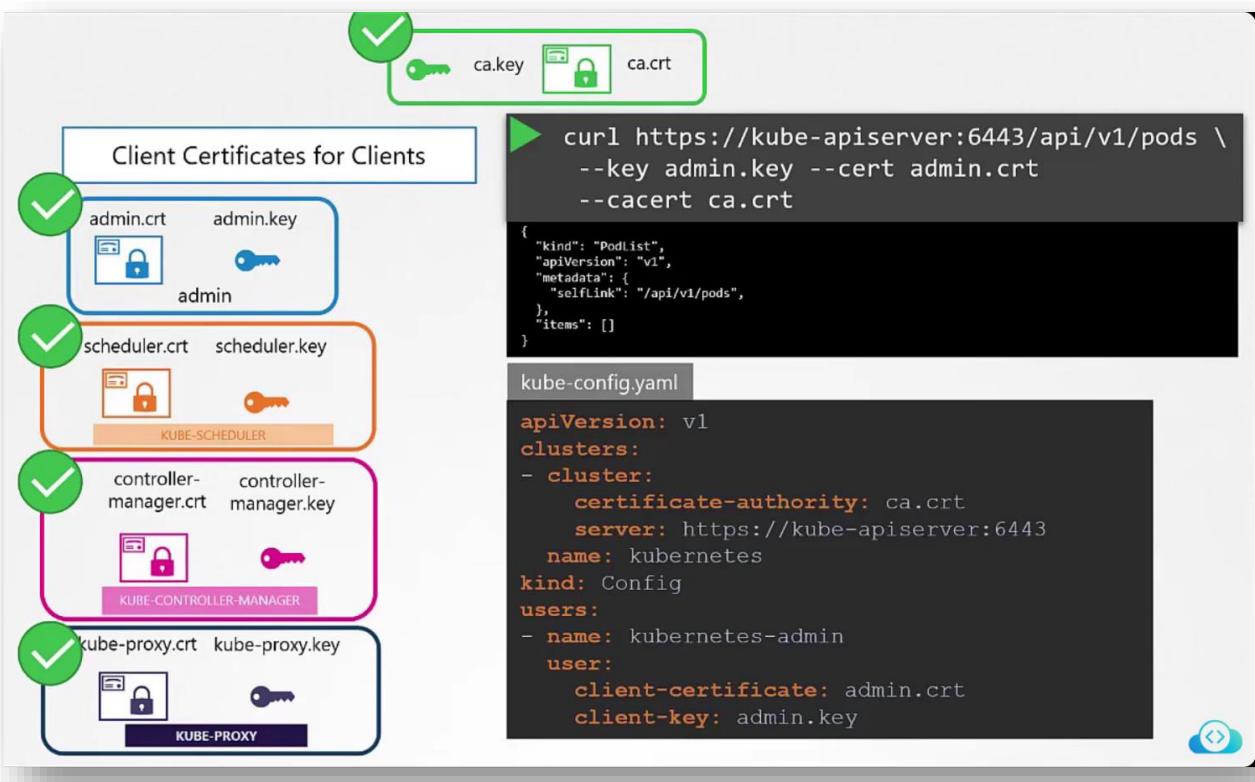
controller-manager.crt



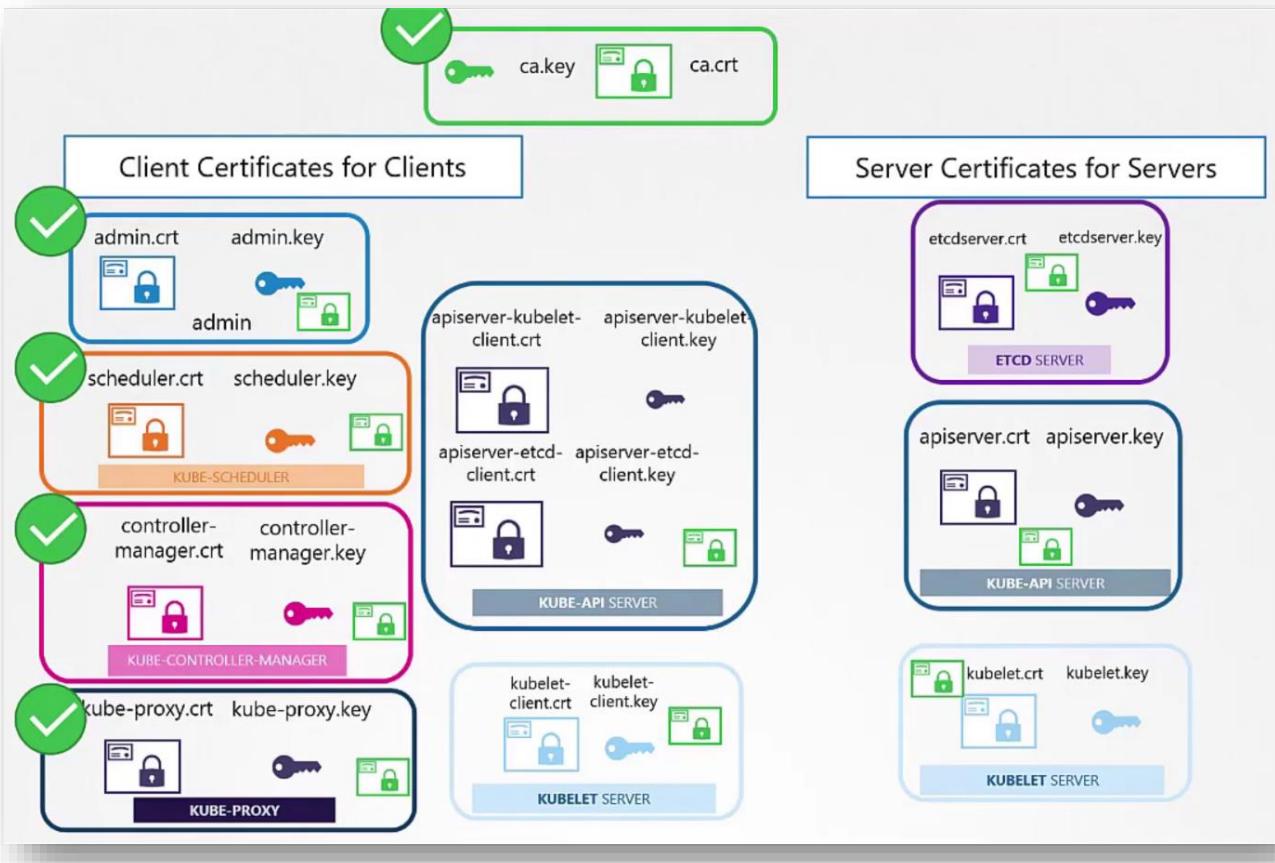
Step 6.



Step 7.



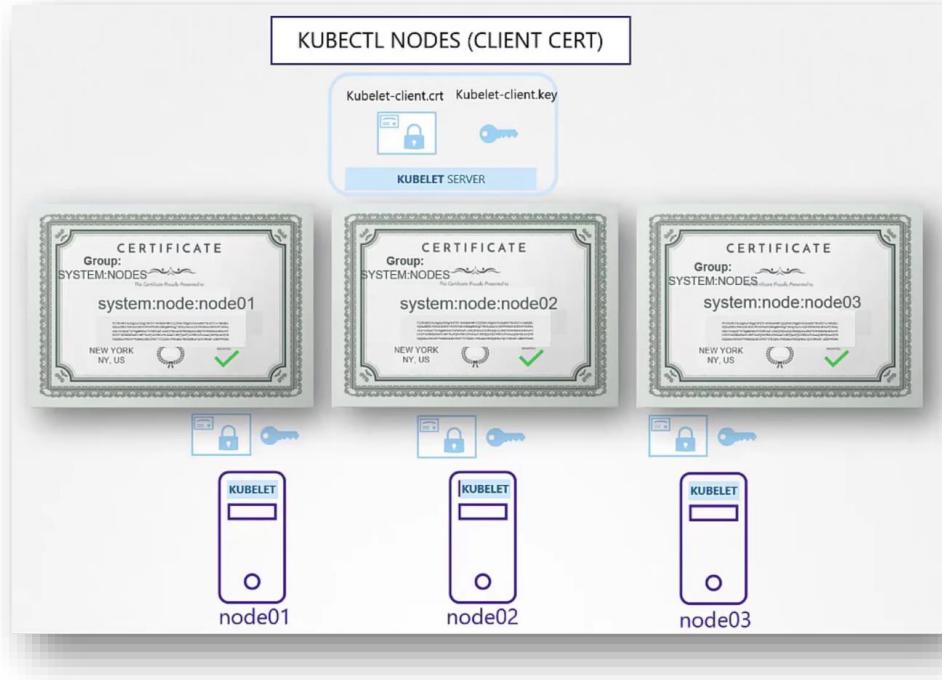
Step 8.



Step 9.



Step 10.

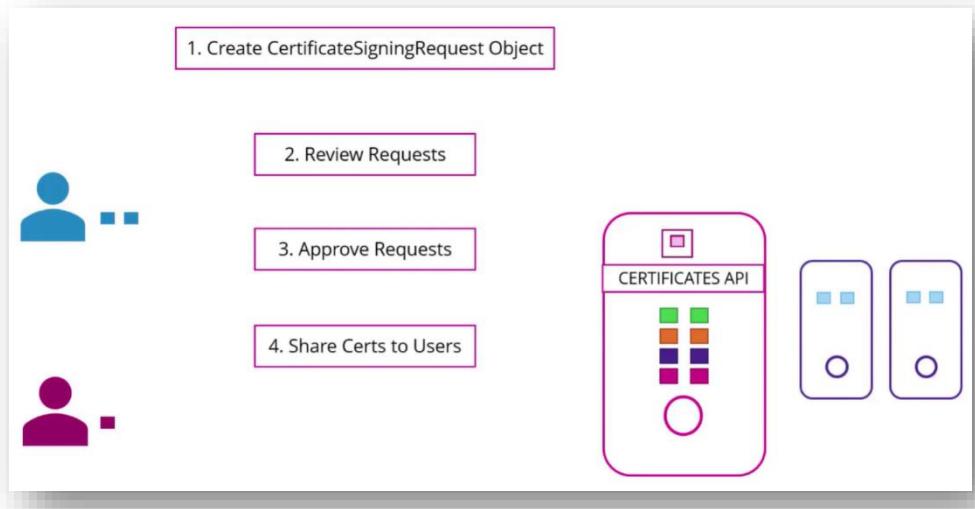


7.8.1 COMMANDLINE

- 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-----
MIICDCCAJAeCAQAwEzERMA0GAlUEAwIBmV3LXVzZXIwgEIMAOGCSqGSIb3DQE
AQAAATBdwIwgExKA0IBAQD0W7W+DXsAJSrJpho5vR1Bp1nZg+6xcs9+UwKK10
LfC27t+1eErON5Muq99NevwME0nrDUO/thyVqfJwz2X0ILDRxJyy+40FbeO+SziyCK
9w08AQsFAOCQAQEAS9156CluxTuF5BBYSU7QFHuz1NxAdYsaORRQnwfLwhG14
hOK4a2zyfy144001jya6tUmBD5xxr8BLK8Kg3srRETjQ15rLzy9LRVsJgh04gY
P9NL+@RSxIDOVsQBaB2nWeypMcJ5Tf53Le+NSNMLQ2++RMn1DQ7juPEic8/dhk
Wz2EUMGUawzykdrlImwvZmIMYR.DHtVY1e+0f9/YElt+FSGjh5L5VuT1Dqiy
413E/y3qL71WFAcuH3OsVpU0nQTSMdsoqKcbsE56CC5DhPGZIpUbnuKUpAwka+8E
vwQ87jG+hpknxmuAEXgIuodAlJ77ju/TD1cw==
-----END CERTIFICATE REQUEST-----





jane.csr

```
-----BEGIN CERTIFICATE REQUEST-----  

MIICDCCAJAeCAQAwEzERMA0GAlUEAwIBmV3LXVzZXIwgEIMAOGCSqGSIb3DQE  

AQAAATBdwIwgExKA0IBAQD0W7W+DXsAJSrJpho5vR1Bp1nZg+6xcs9+UwKK10  

LfC27t+1eErON5Muq99NevwME0nrDUO/thyVqfJwz2X0ILDRxJyy+40FbeO+SziyCK  

9w08AQsFAOCQAQEAS9156CluxTuF5BBYSU7QFHuz1NxAdYsaORRQnwfLwhG14  

hOK4a2zyfy144001jya6tUmBD5xxr8BLK8Kg3srRETjQ15rLzy9LRVsJgh04gY  

P9NL+@RSxIDOVsQBaB2nWeypMcJ5Tf53Le+NSNMLQ2++RMn1DQ7juPEic8/dhk  

Wz2EUMGUawzykdrlImwvZmIMYR.DHtVY1e+0f9/YElt+FSGjh5L5VuT1Dqiy  

413E/y3qL71WFAcuH3OsVpU0nQTSMdsoqKcbsE56CC5DhPGZIpUbnuKUpAwka+8E  

vwQ87jG+hpknxmuAEXgIuodAlJ77ju/TD1cw==  

-----END CERTIFICATE REQUEST-----
```

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:
    LS0tLS1CRUdJTIBDRVJUSUZJQ0FURSBRSVFVRVNULS0
    tLS0tKTU1JQ1dEQ0NBVUFDQVFBD0V6RVJNQThHQTFVFR
    F3d0libVVzTFhwelpSXdnZ0VpTEwR0NtCudTSWIzR
    FFFQgpBUVVBQTRJQkR3QxdnZ0VLQW9JQkFRRE8wV0
    K0RYc0FKU01yanB0bzV2Uk1CcGxuemcrNnhj0StVvnd
    rS2kwCkxmQzI3dCsxZUVuT041TXv0tLoZXztTUVPbn
    J
```

```
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
```

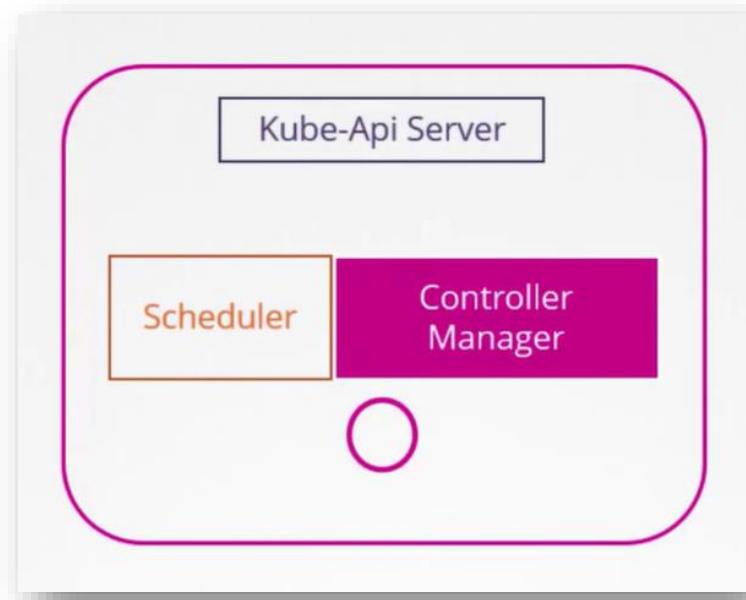
jane approved!

```
▶ 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
Q2wxYXoxawL5M3JNVisreFRYQUowJ3ndn0RRwUpLb1pJaHZjTkFRRUwKQ1FBd0ZURVRN
QkVHQTFVRUF4TUthM1ZpWlhKdVpUYmxjekF1RncweE9UQX1NVE14TmpNeU1EQmFGd1dn
Y8ZFeD1ajNuSXY3eFdS1IRn5su041c0t5Z0vxUkwzTFMSV29GelhHZDdwCmlEZ2FO
MVVRMFbxTVhjN09FVnVjsWc1Yk4weEVHTkVwRUsdU1BN1zWeHVjS1h6aG91dDY0MED1
MGU0YXFkWVIKwmVmMbjBvRTFCY3dod2xic0I1ND0kLS0tLS1FTkQgQ0eVSVE1gsUNBVfUt
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-----
MIICWDCCAUACQAWEzERMA8GA1UEAwIBmV3LXvZXIWg
AQUAA4IBDwAwggEKAoIBAQD00WJW+DXsAJSIrjpNo5vRIE
Lfc27t+1eEnoN5Muq99NevmMEOnrDUO/thyVqp2w2XNIDR
y3BihhB93MJ70ql3UTvZ8TELqyaDknR1/jv/SxgXkok0AB
IF5nxAttMVkDPQ7NbeZRG43b+QW1VGR/z6DW0fJnbfez0t
EcCXAwqChjBLkz2BHP4J89D6Xb8k39pu6jpyngV6uP0tI
j2qEl+hZEWkkFz801NNtyT5LxMqENDCnIgwC4Gz1RGbrAg
9w0BAQsFAAOCAQEAS9iS6C1uxTuF5BBYSU7QFQHuzalNx
hOK4a2zyNy14400ijyaD6tUW8DSxkr8BLK8Kg3srEtJqI
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF53lesNSNMLQ2++R
Wr2EUM6UawzykrdHImwTv2m1MY0R+DntV1Yie+0H9/YEl
4l3E/y3qL71WfAcuH3OsVpUUnQISMdQs0qWcsbE56CC5Df
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.



```
▶ curl https://my-kube-playground:6443/api/v1/pods \
  --key admin.key
  --cert admin.crt
  --cacert ca.crt
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/pods",
  },
  "items": []
}
```



```
▶ kubectl get pods
  --server my-kube-playground:6443
  --client-key admin.key
  --client-certificate admin.crt
  --certificate-authority ca.crt
No resources found.
```

Step 2.

\$HOME/.kube/config

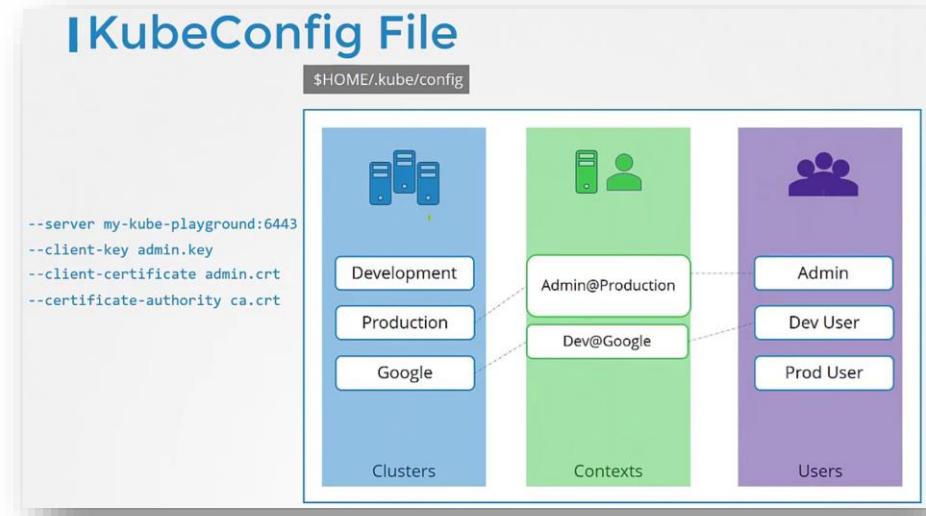
KubeConfig File

```
--server my-kube-playground:6443  
--client-key admin.key  
--client-certificate admin.crt  
--certificate-authority ca.crt
```

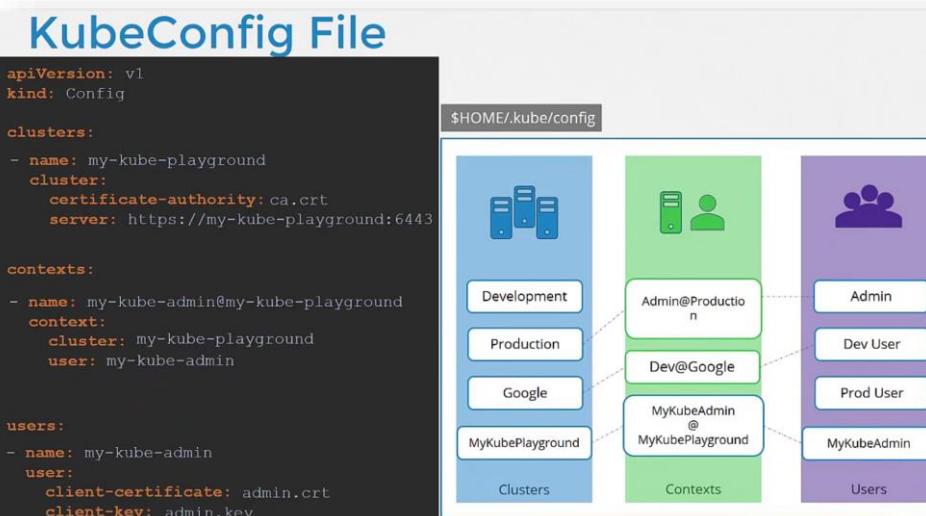
kubectl get pods

No resources found.

Step 3.



Step 4.



Step 5.

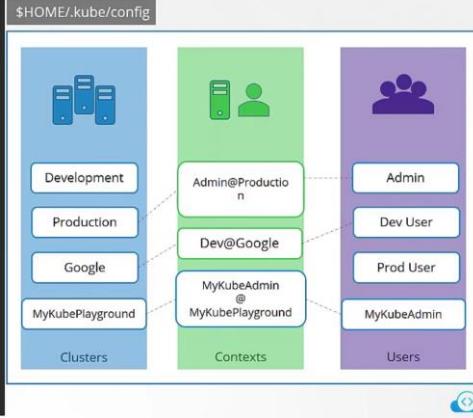
KubeConfig File

```
apiVersion: v1
kind: Config
current-context: dev-user@google

clusters:
- name: my-kube-playground  (values hidden...)
- name: development
- name: production
- name: google

contexts:
- name: my-kube-admin@my-kube-playground
- name: dev-user@google
- name: prod-user@production

users:
- name: my-kube-admin
- name: admin
- name: dev-user
- name: prod-user
```



Step 6.

| Kubectl config

```
▶ kubectl config view
apiVersion: v1
kind: Config
current-context: my-kube-admin@my-kube-playground

clusters:
- name: my-kube-playground
- name: development
- name: production

contexts:
- name: my-kube-admin@my-kube-playground
- Name: prod-user@production

users:
- name: my-kube-admin
- name: prod-user
```

```
▶ kubectl config use-context prod-user@production
apiVersion: v1
kind: Config
current-context: prod-user@production

clusters:
- name: my-kube-playground
- name: development
- name: production

contexts:
- name: my-kube-admin@my-kube-playground
- Name: prod-user@production

users:
- name: my-kube-admin
- name: prod-user
```

Step 7.

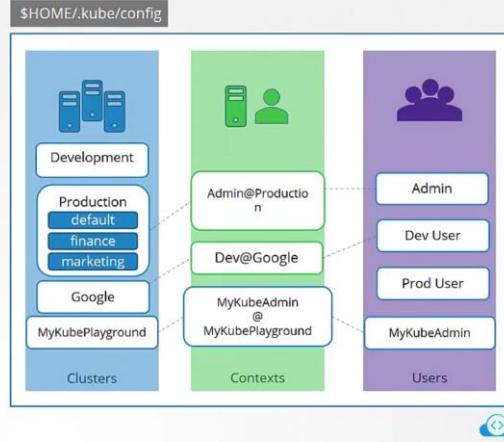
| Namespaces

```
apiVersion: v1
kind: Config

clusters:
- name: production
  cluster:
    certificate-authority: ca.crt
    server: https://172.17.0.51:6443

contexts:
- name: admin@production
  context:
    cluster: production
    user: admin
    namespace: finance

users:
- name: admin
  user:
    client-certificate: admin.crt
    client-key: admin.key
```



Step 8.

Certificates in KubeConfig

```
apiVersion: v1
kind: Config

clusters:
- name: production
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://172.17.0.51:6443

contexts:
- name: admin@production
  context:
    cluster: production
    user: admin
    namespace: finance

users:
- name: admin
  user:
    client-certificate: /etc/kubernetes/pki/users/admin.crt
    client-key: /etc/kubernetes/pki/users/admin.key
```

Step 9.

Certificates in KubeConfig

```
apiVersion: v1
kind: Config

clusters:
- name: production
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt

certificate-authority-data: LS0tLS1CRUdJTiBDRVJU
SUZQOFURSBRSRVFRVNRNLS0tLS0KTU1J
Q1dEQ0NBVFUDQVFBD0V6RVJNQThHQTFV
RUF3d0libVYzTFhWelpYSXdnZ0VpTUEw
RONc0dTSWIzRFFFQgpBUVVBQTRjQkr3
QXdnZ0VLQW9JQkFRRRE8wV0pXKORYc0FK
U0lyanB0bzV2Uk1CcGxuemcrNnhj0StV
VndrS2kwCkxmQzI3dCsxZUVuT041TXvx
OT1OZXztTUVPbnJ

-----BEGIN CERTIFICATE-----
MIICWCCAUACAgAwEzEMARBgA1UEAwIBmI3LXvZXInggEiMA0G
AQAA4iB0AwggEKAoIBAQD00WjW+DXSAJS1rjpNo5vR1BpInzg+
Lfc27+1eN0NSMuq9NevvCOnr0U0/thyPgPz2XNIDRXyYf4
y3b1hB93kJ70l3UTv8TElyadkn1/Jy/SxgXkok9ABUTpMw
IFnxAttCMkPQ7hbzR643bQn1VGK/260wfnbfezotaAydu
EcxXmQchjlkzZBHP4J8906X8k39pu6pyngV6Up0tb0zpqh
j2qELhZEWKKF2801NTy15LXhQEND1igc4Gz1GbAgMBAAG/
9wBAQsFAOCQAE59156C1uxIuf5BBY5U7QfQfUzaInNxAdys0H
hOK4z2zyLy144001jya6stUW05krBLKk3sREtNq15rL7y9
P9Nl+oDR5xR0V5QbaB2nWeypM5cJ5Tf531sSNMLQ2+rRmjQ07
W2EU6GuazykrhdImw7v2mMYR8-DHtV1yie+0H9/VE1+fSgjhs
413E/yqL71kfacuH3OsVpUUmQZShdg8qjcsbe56CCSDhPGZ1put
vNQ87JGhpknxmuFAeXxgtwodAlaJ7ju/TDlcw==
-----END CERTIFICATE-----
> cat ca.crt | base64
```

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.

The image shows two terminal windows side-by-side. The left window displays the command `curl https://kube-master:6443/version` followed by its JSON output, which includes details about the Kubernetes version (v1.13.0), git commit, build date, go version, compiler, and platform. The right window displays the command `curl https://kube-master:6443/api/v1/pods` followed by its JSON output, which lists a single pod named "nginx-5c7588df-ghsb" with its metadata, labels, and owner references.

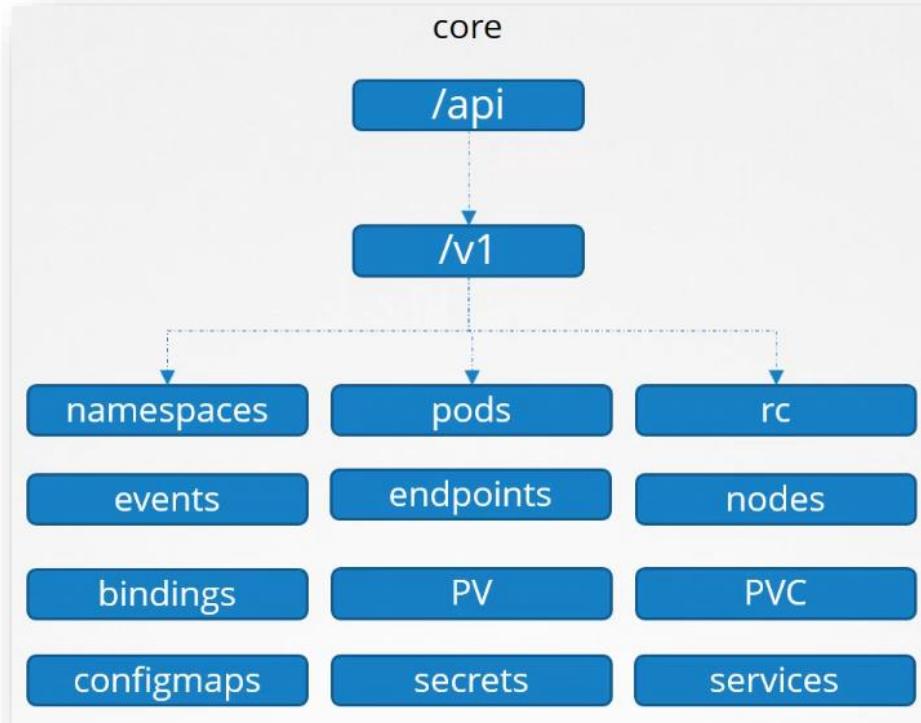
```
curl https://kube-master:6443/version
{
  "major": "1",
  "minor": "13",
  "gitVersion": "v1.13.0",
  "gitCommit": "dd47ac13c1a9483ea035a79cd7c10005ff21a6d",
  "gitTreeState": "clean",
  "buildDate": "2018-12-03T20:56:12Z",
  "goVersion": "go1.11.2",
  "compiler": "gc",
  "platform": "linux/amd64"
}

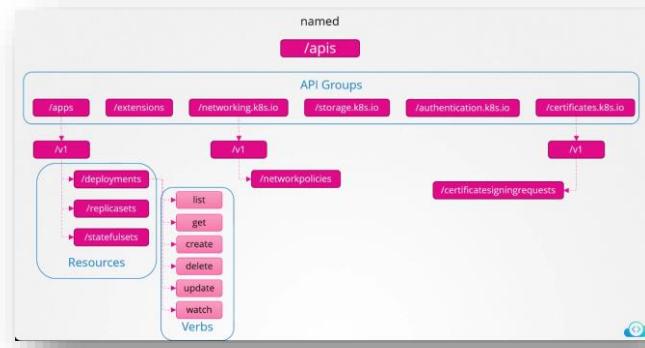
curl https://kube-master:6443/api/v1/pods
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/pods",
    "resourceVersion": "153068"
  },
  "items": [
    {
      "metadata": {
        "name": "nginx-5c7588df-ghsb",
        "generateName": "nginx-5c7588df-",
        "namespace": "default",
        "creationTimestamp": "2019-03-20T10:57:48Z",
        "labels": {
          "app": "nginx",
          "pod-template-hash": "5c7588df"
        },
        "ownerReferences": [
          {
            "apiVersion": "apps/v1",
            "kind": "ReplicaSet",
            "name": "nginx-5c7588df",
            "uid": "398ce179-4af9-11e9-beb6-020d3114c7a7",
            "controller": true,
            "blockOwnerDeletion": true
          }
        ],
        "resourceVersion": "153068"
      }
    }
  ]
}
```

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",
  ]
}

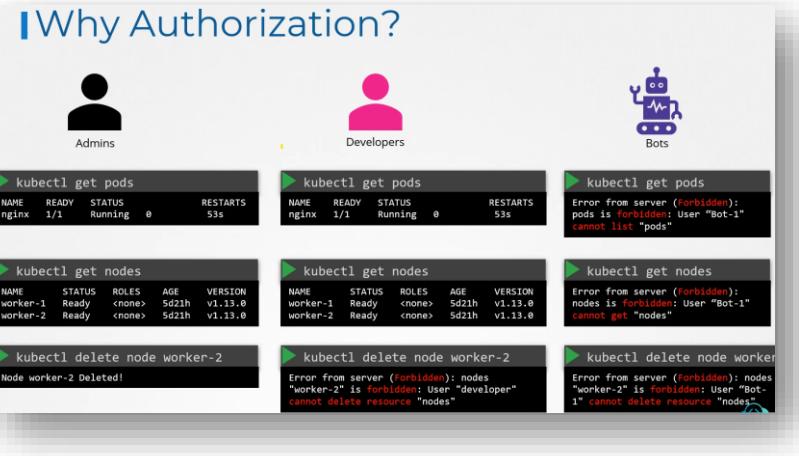
curl http://localhost:6443/apis -k | grep "name"
{
  "name": "extensions",
  "name": "apps",
  "name": "events.k8s.io",
  "name": "authentication.k8s.io",
  "name": "authorization.k8s.io",
  "name": "autoscaling",
  "name": "batch",
  "name": "certificates.k8s.io",
  "name": "networking.k8s.io",
  "name":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

kubectl proxy



Kube proxy **Kubectl proxy**

7.13 AUTHORIZATION

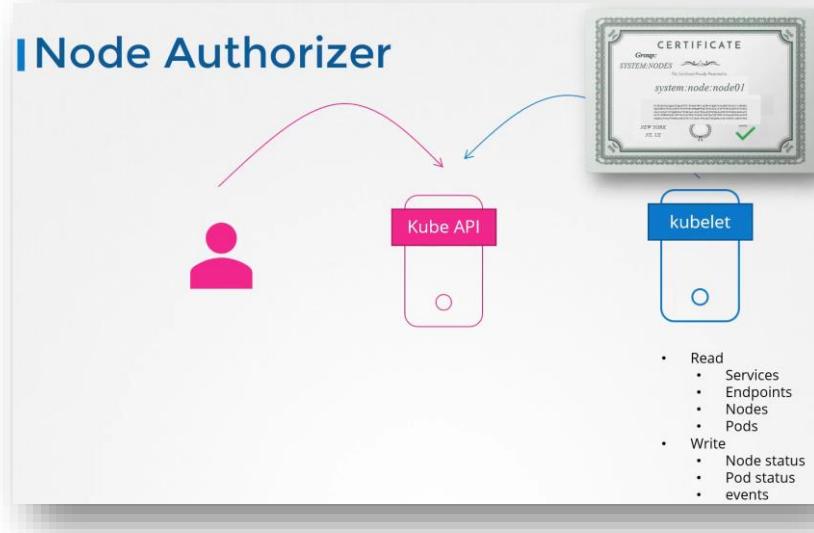


7.13.1 TYPES OF AUTHORIZATION

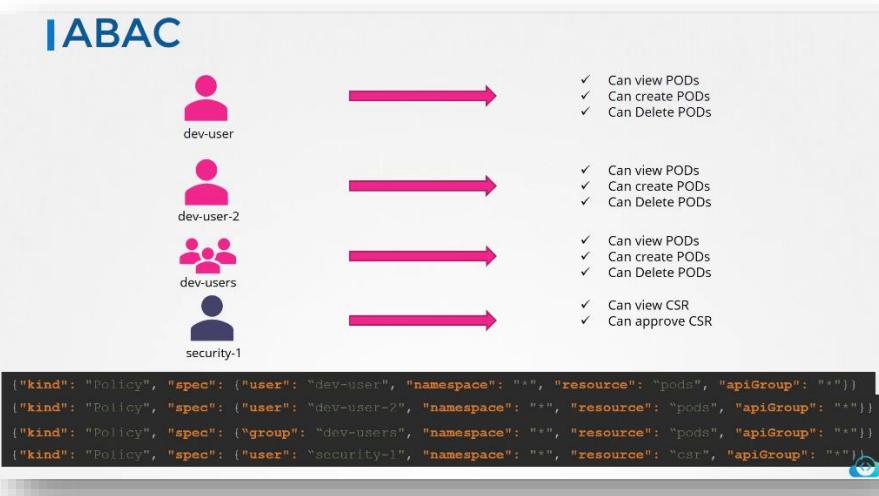
- Node
- ABAC (Attributes base authorization)
- RBAC (Roll base Authorization)
- Webhook



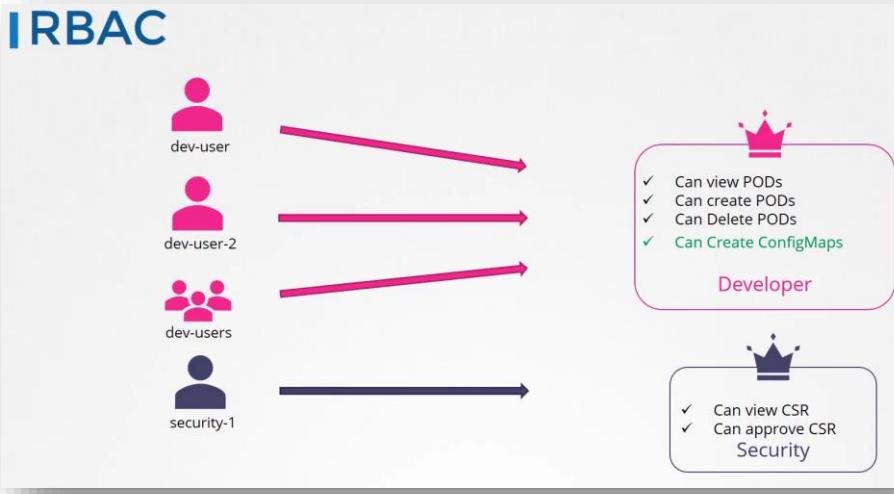
Step 13.



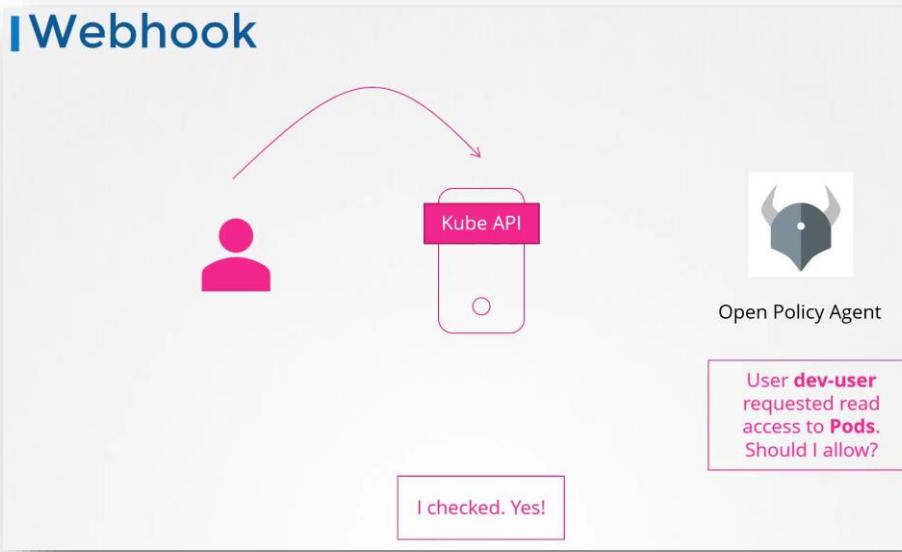
Step 15.



Step 16.



Step 17.



Step 18.

I Authorization Mode

NODE

ABAC

RBAC

WEBHOOK

AlwaysAllow

AlwaysDeny

Step 19.

I Authorization Mode

AlwaysAllow

NODE

ABAC

RBAC

WEBHOOK

AlwaysDeny

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

Step 20.

I Authorization Mode

AlwaysAllow

NODE

ABAC

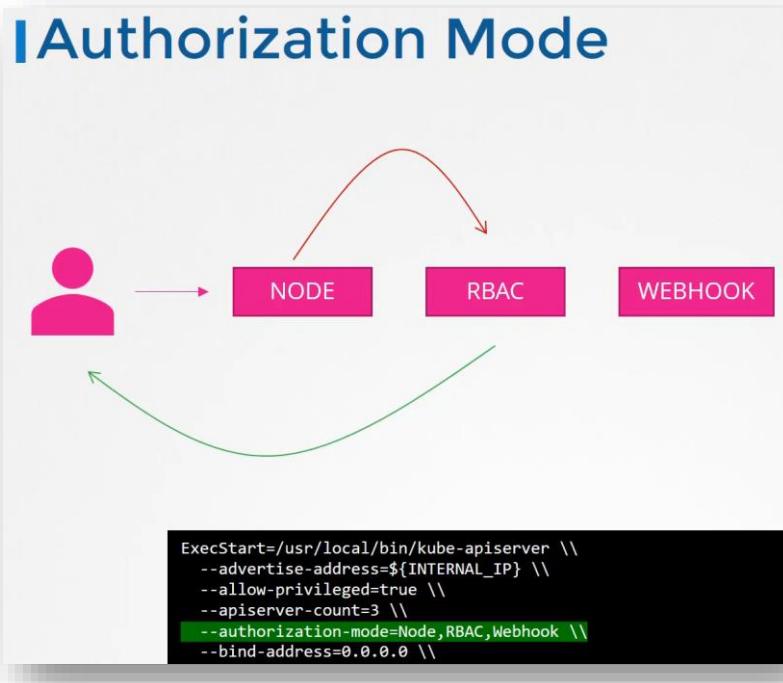
RBAC

WEBHOOK

AlwaysDeny

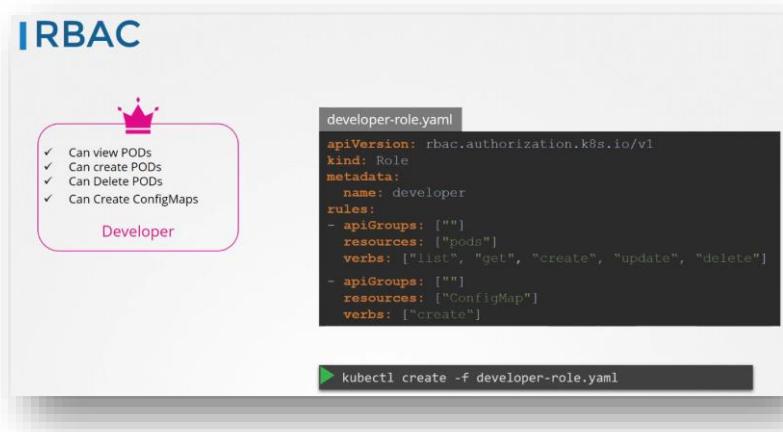
```
ExecStart=/usr/local/bin/kube-apiserver \
--advertise-address=${INTERNAL_IP} \
--allow-privileged=true \
--apiserver-count=3 \
--authorization-mode=Node,RBAC,Webhook \
--bind-address=0.0.0.0 \
--enable-swagger-ui=true \
--etcd-cafile=/var/lib/kubernetes/ca.pem \
--etcd-certfile=/var/lib/kubernetes/apiserver-etcd-client.crt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

Step 21.

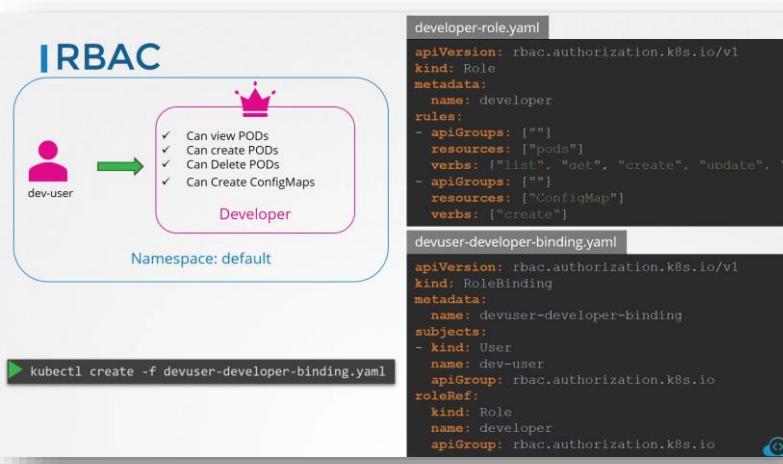


7.14 RBAC (ROLE BASED ACCESS CONTROLS)

Step 22.



Step 23.



Step 24.

IView RBAC

```
▶ kubectl get roles
NAME      AGE
developer  4s

▶ kubectl get rolebindings
NAME                AGE
devuser-developer-binding  24s

▶ kubectl describe role developer
Name:      developer
Labels:    <none>
Annotations: <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  -----      -----           -----          -----
  ConfigMap []              []            [create]
  pods       []              []            [get watch list create delete]
```

Step 25.

IView RBAC

```
▶ kubectl describe rolebinding devuser-developer-binding
Name:      devuser-developer-binding
Labels:    <none>
Annotations: <none>
Role:
  Kind:  Role
  Name:  developer
Subjects:
  Kind  Name      Namespace
  ----  --        --
  User  dev-user
```

Step 26.

ICheck Access

```
▶ kubectl auth can-i create deployments
yes

▶ kubectl auth can-i delete nodes
no

▶ kubectl auth can-i create deployments --as dev-user
no

▶ kubectl auth can-i create pods --as dev-user
yes

▶ kubectl auth can-i create pods --as dev-user --namespace test
no
```

Resource Names



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`

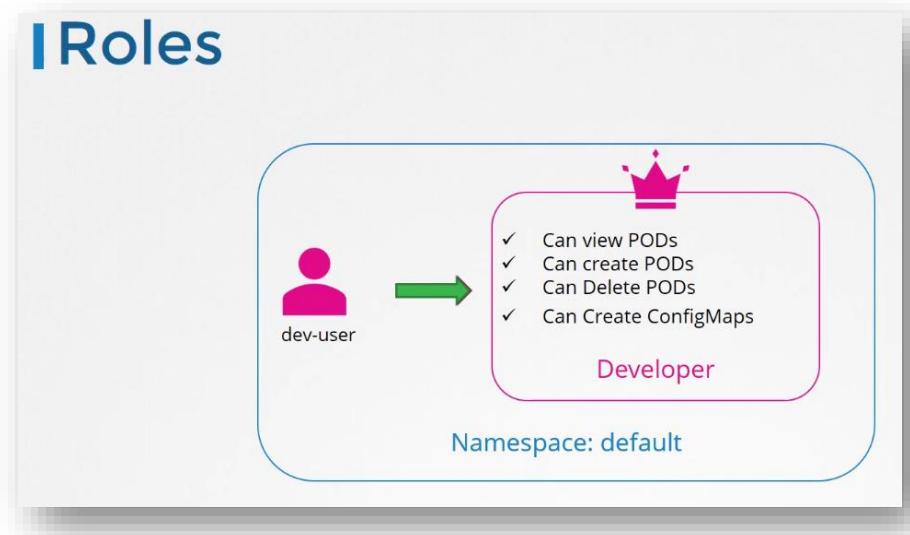
- `kubectl create role developer --namespace=default --verb=list,create,delete --resource=pods`

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

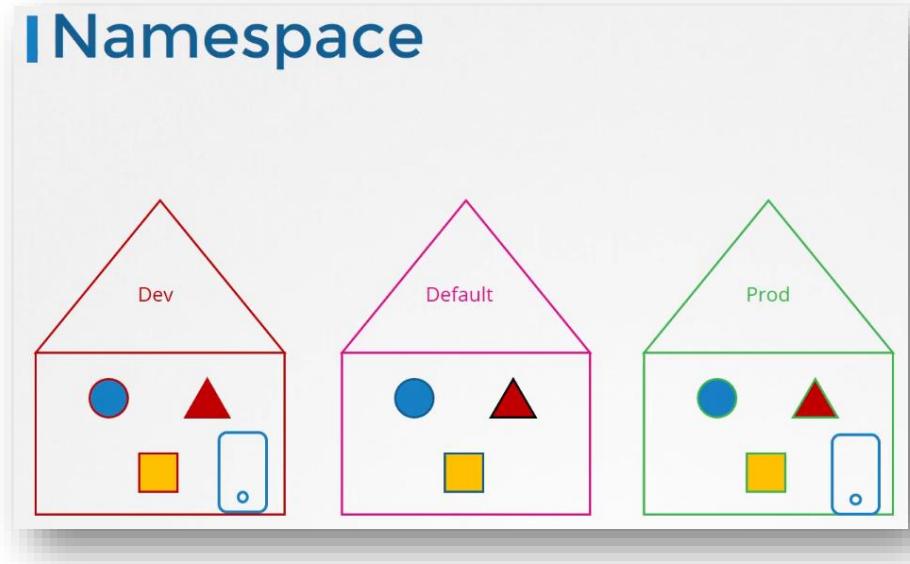
- `kubectl edit role developer -n blue`
- `kubectl get role -o wide -A`
- `cat /etc/kubernetes/manifests/api-server.yaml`
- `kubectl get roles -A -no-headers | wl -l`
- `kubectl get rolebindings -n kube-system`
- `kubectl get pods --as dev-user`
- `kubectl create role --help`

7.15 CLUSTER ROLES

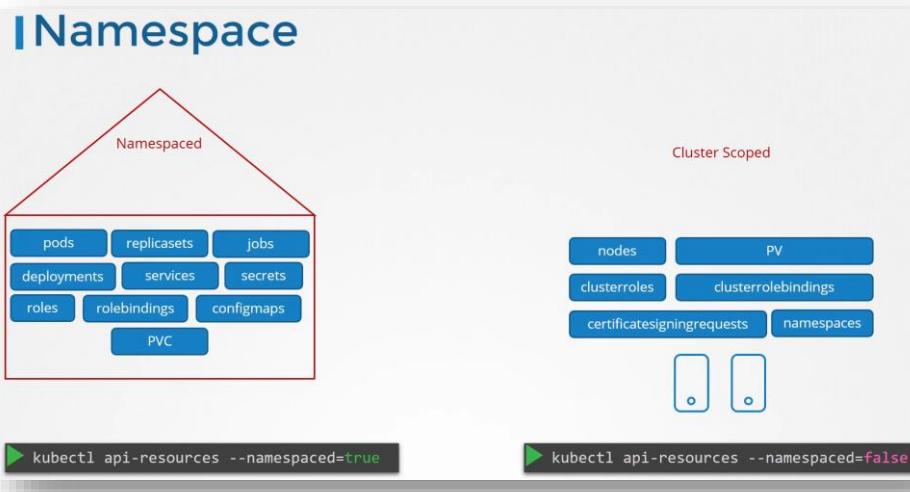
Step 28.



Step 29.



Step 30.



Step 31.

clusterroles

Cluster Admin

- ✓ Can view Nodes
- ✓ Can create Nodes
- ✓ Can delete Nodes

Storage Admin

- ✓ Can view PVs
- ✓ Can create PVs
- ✓ Can delete PVCs

cluster-admin-role.yaml

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

kubectl create -f cluster-admin-role.yaml

Step 32.

clusterrolebinding

cluster-admin

Cluster Admin

- ✓ Can view Nodes
- ✓ Can create Nodes
- ✓ Can delete Nodes

cluster-admin-role.yaml

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

cluster-admin-role-binding.yaml

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

Step 33.

```
---
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.

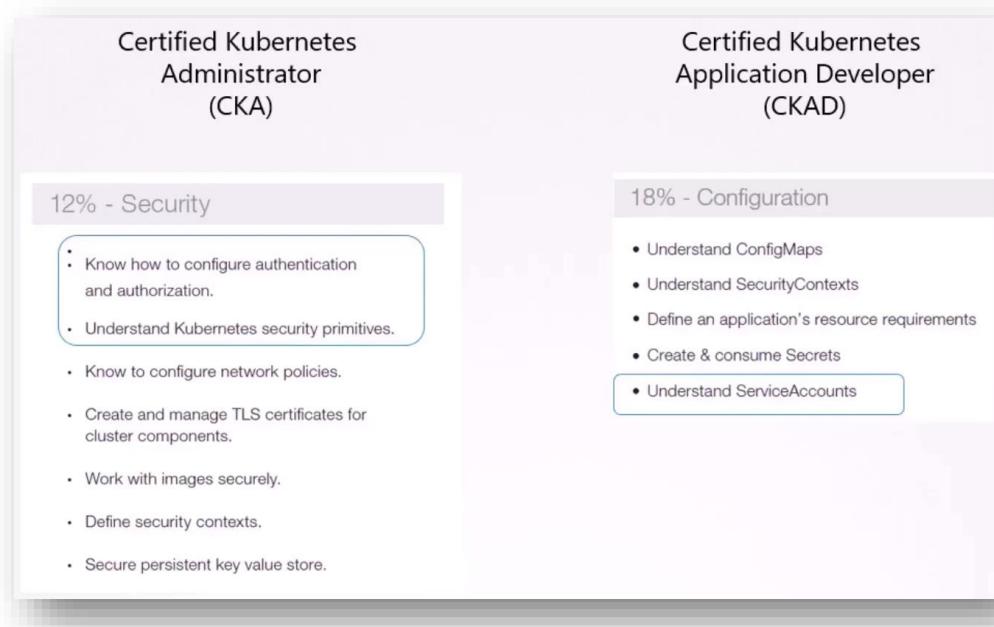
```
---  
kind: ClusterRole  
apiVersion: rbac.authorization.k8s.io/v1  
metadata:  
  name: storage-admin  
rules:  
  - apiGroups: [""]  
    resources: ["persistentvolumes"]  
    verbs: ["get", "watch", "list", "create", "delete"]  
  - apiGroups: ["storage.k8s.io"]  
    resources: ["storageclasses"]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

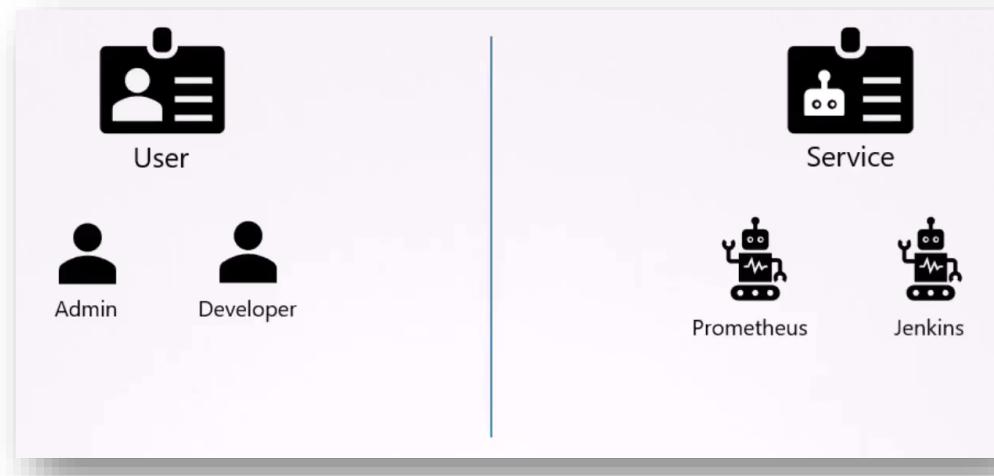
- kubectl get clusterroles --no-headers | wc -l
(-c=Count bytes, -m=Count characters, -l=Count newlines, -w= Count words
-L=Print longest line length)
- kubectl get clusterrolebindings --no-headers | wc -l
- kubectl api-resources --namespaced=false
- kubectl describe clusterrolebinding cluster-admin
- kubectl describe clusterrole cluster-admin
- kubectl clusterrolebindings | grep cluster-admin
- kubectl get nodes –as michelle
- kubectl create clusterrole --help
- kubectl create clusterrole michelle-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

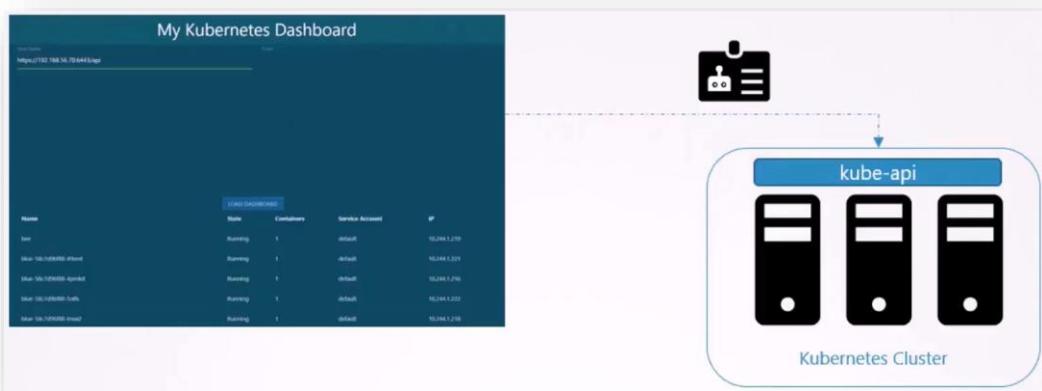
Step 35.



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.

```
▶ 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>
```



```
▶ kubectl describe secret dashboard-sa-token-kbbdm
Name:           dashboard-sa-token-kbbdm
Namespace:      default
Labels:          <none>
Type:           kubernetes.io/service-account-token
Data
=====
ca.crt:    1025 bytes
namespace:  7 bytes
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlc5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcyc5pby9zZXJ2aWNlYWNgb3Vud...3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcyc5pby9zZXJ2aWNlYWNgb3Vud...
```




Secret

token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlc5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcyc5pby9zZXJ2aWNlYWNgb3Vud...

Step 40.

```
▶ curl https://192.168.56.70:6443/api --insecure
--header "Authorization: Bearer eyJhbG..."
```




Secret

token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlc5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcyc5pby9zZXJ2aWNlYWNgb3Vud...

Step 41.

```
▶ kubectl describe pod my-kubernetes-dashboard
Name:           my-kubernetes-dashboard
Namespace:      default
Annotations:    <none>
Status:         Running
IP:             10.244.0.15
Containers:
  nginx:
    Image:        my-kubernetes-dashboard
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-j4hkv (ro)
    Conditions:
      Type     Status
    Volumes:
      default-token-j4hkv:
        Type:       Secret (a volume populated by a Secret)
        SecretName: default-token-j4hkv
        Optional:   false

▶ kubectl exec -it my-kubernetes-dashboard -- ls /var/run/secrets/kubernetes.io/serviceaccount
ca.crt  namespace  token

▶ kubectl exec -it my-kubernetes-dashboard cat /var/run/secrets/kubernetes.io/serviceaccount/token
eyJhbGciOiJSUzI1NiIsImtpZC16IiI9.eyJpc3MiOiJrdWJ1cm5ldGVzL3NlcnZpY2VhY2Nwdm50Iiwia3V1ZXJuZXrlcy5pb9zZXJ2awN1YwNjb3VudC9uYwI1c3BhY2U1O1JkZwdxhIiwi3V1ZXJuZXRLcy5pb9zZXJ2aNN1YwNjb3VudC9zZwRyZXQubmf75161m1zmf1bHQtG9rZw4tajRoa3YilCJrdWJ1cm51dgVzLmlvL3NlcnZpY2hY2Nwdm50L3NlcnZpY2UEYwNjb3VudCsuyW1LTjo1ZGmXVsDCtsImt1YmVybmv0ZxNuahSv2VydmljZMFjY291bnQvc2VydmljZS1hY2Nwdm50LnVpZC16IjcxZGM4YmExLTU2MGMEMTF10C04YmI0LTA4MDayNzkzMTA3MiisInN1Y1161nN5c3R1b1pzZXJ2awN
```

Step 42.

```
▶ kubectl get serviceaccount
NAME      SECRETS   AGE
default   1          218d

▶ pod-definition.yml
apiVersion: v1
kind: Pod
metadata:
  name: my-kubernetes-dashboard
spec:
  containers:
    - name: my-kubernetes-dashboard
      image: my-kubernetes-dashboard

▶ kubectl describe pod my-kubernetes-dashboard
Name:           my-kubernetes-dashboard
Namespace:      default
Annotations:    <none>
Status:         Running
IP:             10.244.0.15
Containers:
  nginx:
    Image:        my-kubernetes-dashboard
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-j4hkv (ro)
    Conditions:
      Type     Status
    Volumes:
      default-token-j4hkv:
        Type:       Secret (a volume populated by a Secret)
        SecretName: default-token-j4hkv
        Optional:   false
```

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.

v1.24

KEP-2799: Reduction of Secret-based Service Account Tokens

```
▶ kubectl create serviceaccount dashboard-sa
serviceaccount "dashboard-sa" created
▶ kubectl create serviceaccount dashboard-sa
serviceaccount "dashboard-sa" created
▶ kubectl create token dashboard-sa
eyJhbGciOiJSUzI1NiIsImtpZCI6I...
```

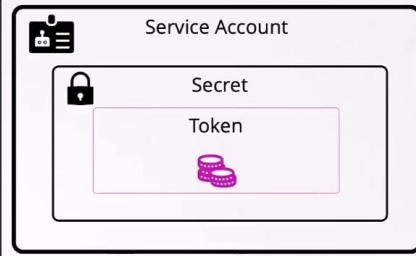
Step 46.

```
▶ jq -R 'split(".") | select(length > 0) | .[0],.[1] | @base64d | fromjson' <<< eyJhbGciOiJSUz...
{
  "alg": "RS256",
  "kid": "v0p_YyzJsnmxNV9uQ8zzs0LIHHA0n3Dy800v1EjXnS5"
}
{
  "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": "7d0fdfc8-fbf3-4ff9-b362-d004297a73f6"
    }
  },
  "nbf": 1664034163,
  "sub": "system:serviceaccount:default:dashboard-sa"
}
```

Step 47.

v1.24

```
secret-definition.yml
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: mysecretname
  annotations:
    kubernetes.io/service-account.name: dashboard-sa
```



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.

Image

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

Step 2.

Image

image: docker.io/library/nginx

The diagram illustrates the structure of a Docker image URL. It shows the string "image: docker.io/library/nginx" above three curly braces. The first brace spans "docker.io", the second spans "library", and the third spans "nginx". Below these braces are three labels: "Registry" under the first brace, "User/Account" under the second, and "Image/Repository" under the third.

Registry User/Account Image/
 Repository

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

Step 3.

Private Repository

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

Login Succeeded
```

Step 4.

Private Repository

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

```
nginx-pod.yaml  
apiVersion: v1  
kind: Pod  
metadata:  
  name: nginx-pod  
spec:  
  containers:  
    - name: nginx  
      image: private-registry.io/apps/internal-app  
imagePullSecrets:  
  - name: regcred
```

```
▶ kubectl create secret docker-registry regcred \  
  --docker-server= private-registry.io \\  
  --docker-username= registry-user \\  
  --docker-password= registry-password \\  
  --docker-email= registry-user@org.com
```

7.17.1 COMMANDLINE

- `kubectl create secret`
- alias `k=kubectl`

7.18 PRE-REQUISITE – SECURITY IN DOCKER

Step 1.

Security

```
▶ docker run ubuntu sleep 3600
```



Step 2.

Security - Users

```
ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
project 3720 0.1 0.1 95580 4916 ?
project 3725 0.0 0.1 95196 4132 ?
project 3727 0.2 0.1 21352 5340 pts/0 S 06:06 0:00 bash
root 3802 0.0 0.0 8924 3616 ?
root 3816 1.0 0.0 4528 828 ?
S 06:06 0:00 docker-containerd-shim -namespace m
root 3816 1.0 0.0 4528 828 ?
S 06:06 0:00 sleep 3600
```

```
ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.0 4528 828 ?
S 03:06 0:00 sleep 3600
```

```
docker run --user=1000 ubuntu sleep 3600
```

```
ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
1000 1 0.0 0.0 4528 828 ?
S 03:06 0:00 sleep 3600
```

Step 3.

Security - Users

```
Dockerfile
FROM ubuntu
USER 1000
```

```
docker build -t my-ubuntu-image .
```

```
docker run my-ubuntu-image sleep 3600
```

```
ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
1000 1 0.0 0.0 4528 828 ?
S 03:06 0:00 sleep 3600
```

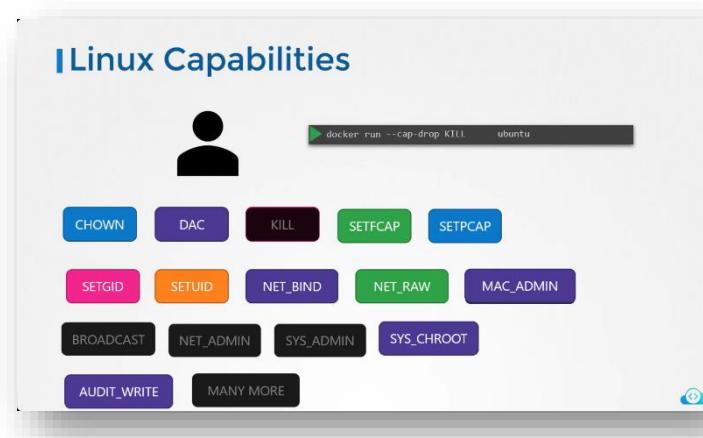
Step 4.

Linux Capabilities

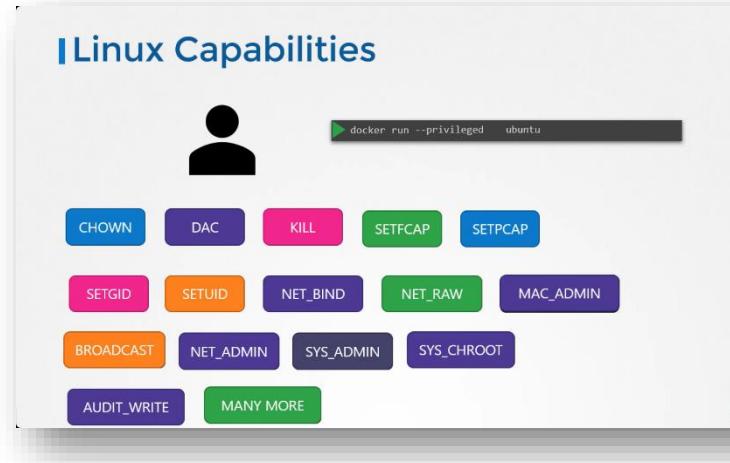
```
docker run --cap-add MAC_ADMIN ubuntu
```

CHOWN	DAC	KILL	SETFCAP	SETPCAP
SETGID	SETUID	NET_BIND	NET_RAW	MAC_ADMIN
BROADCAST	NET_ADMIN	SYS_ADMIN	SYS_CHROOT	
AUDIT_WRITE	MANY MORE			

Step 5.

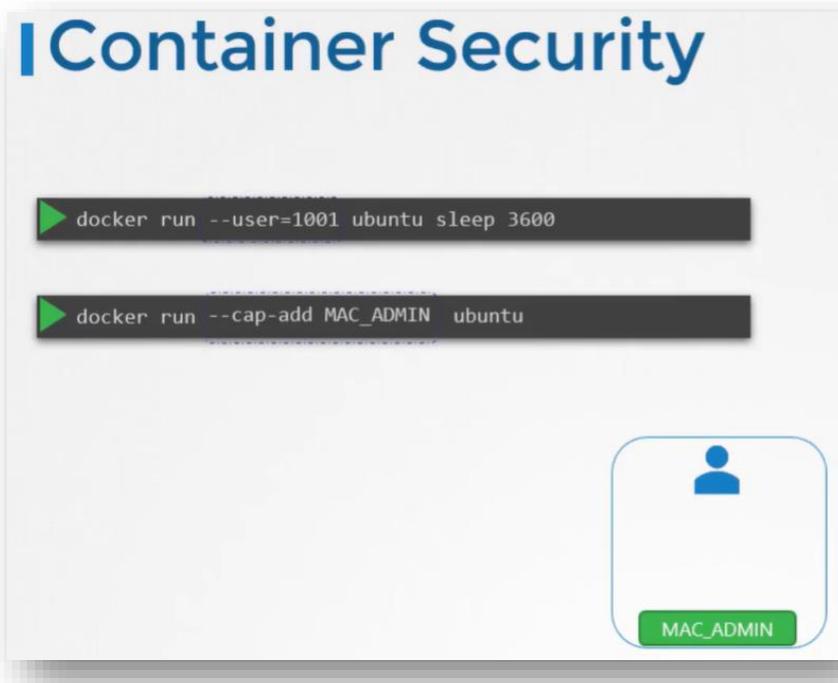


Step 6.



7.19 SECURITY CONTEXTS

Step 7.



Security Context

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



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

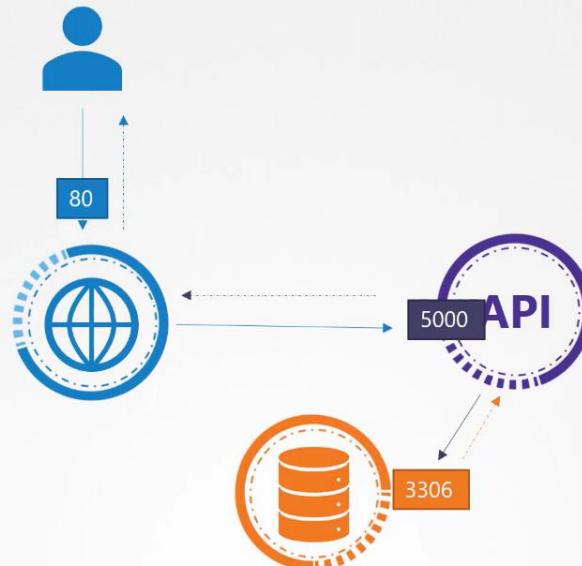


7.19.1 COMMANDLINE

- kubectl exec ubuntu-sleeper – whoami
-

7.20 NETWORK POLICIES

Traffic



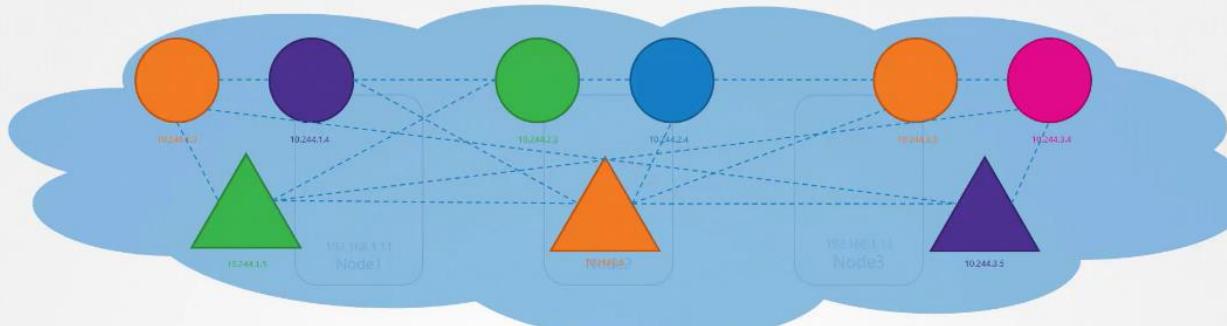
Step 2.

| Traffic



Step 3.

| Network Security



"All Allow"



Step 4.

Network Policy



80

Web
Pod

5000

API
Pod

3306

DB
Pod
Network
Policy



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.

Note

Solutions that Support Network Policies:

- Kube-router
- Calico
- Romana
- Weave-net

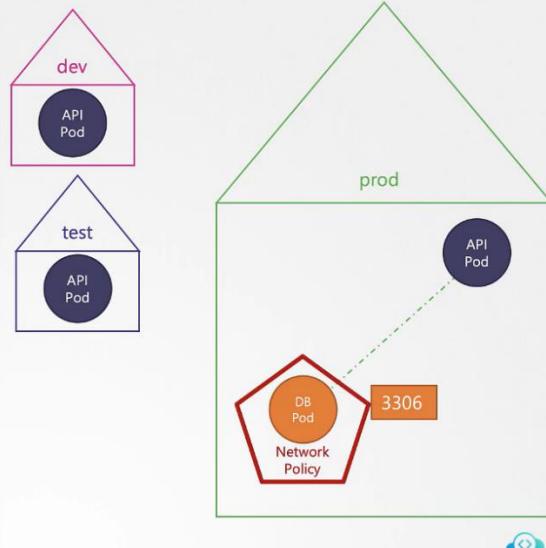
Solutions that DO NOT Support Network Policies:

- Flannel

7.21 DEVELOPING NETWORK POLICIES

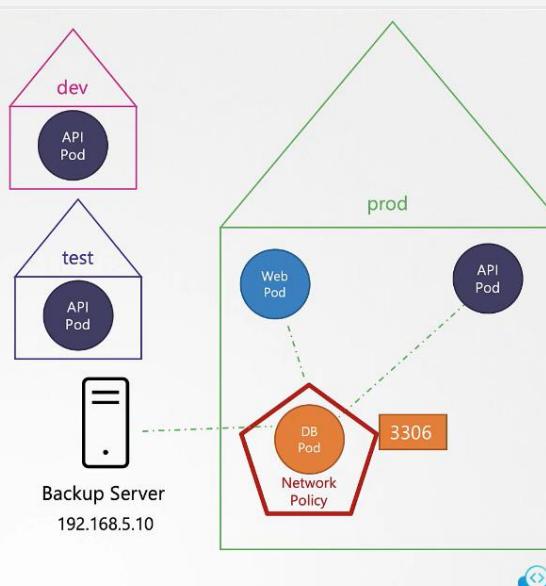
Step 1.

```
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
        namespaceSelector:
          matchLabels:
            name: prod
    ports:
      - protocol: TCP
        port: 3306
```



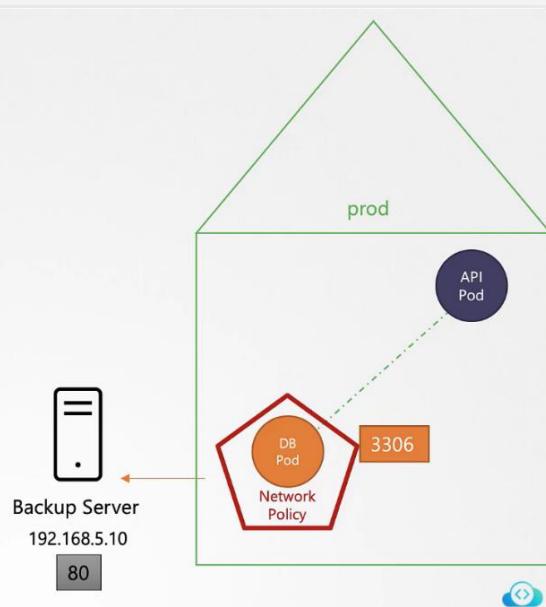
Step 2.

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



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

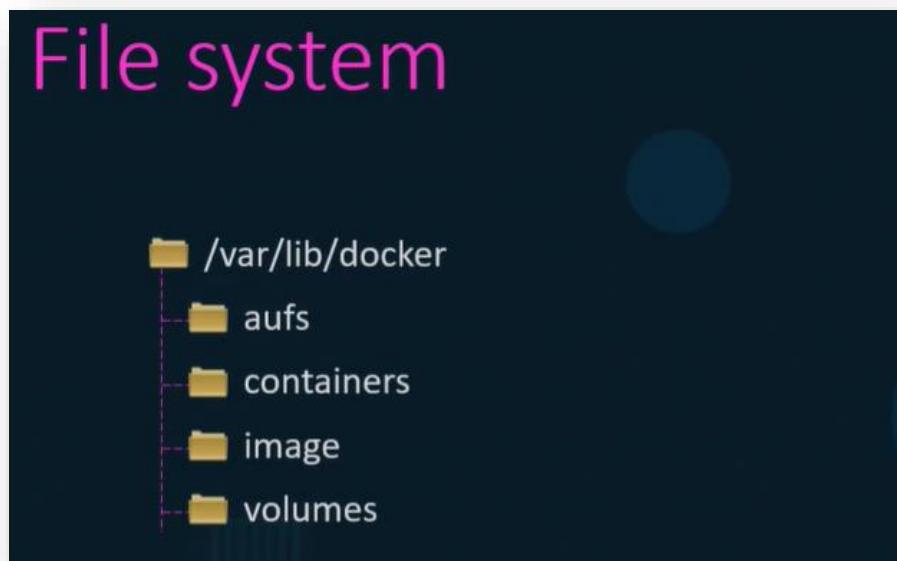
8.2 INTRODUCTION TO DOCKER STORAGE

Step 1.



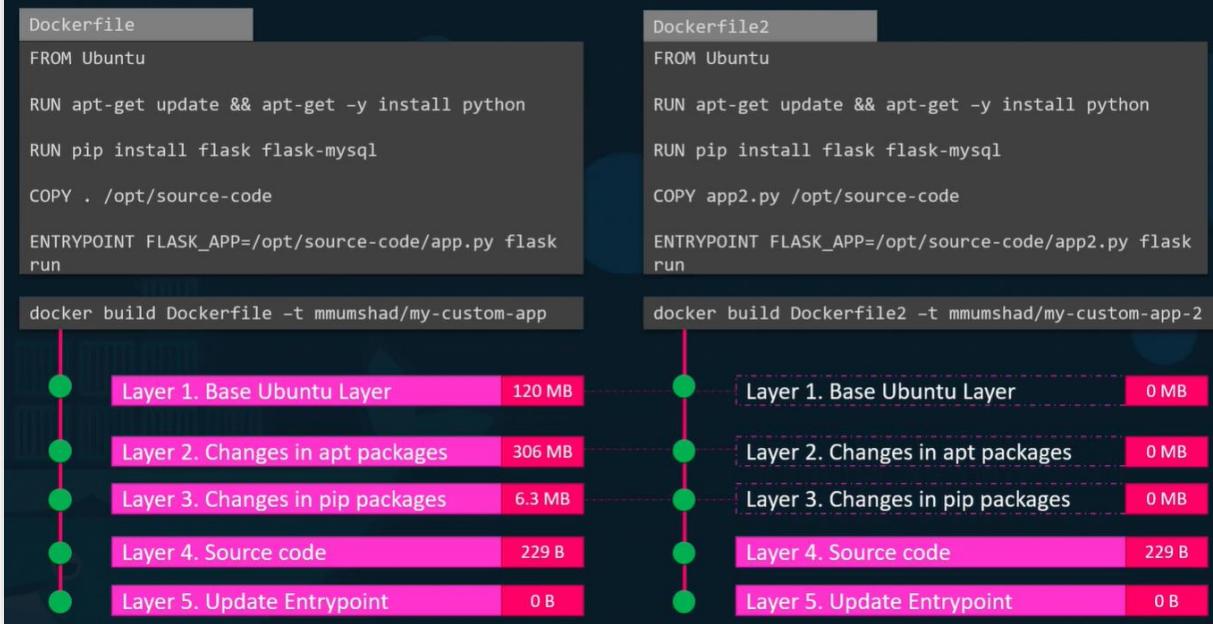
8.3 STORAGE IN DOCKER

Step 1.

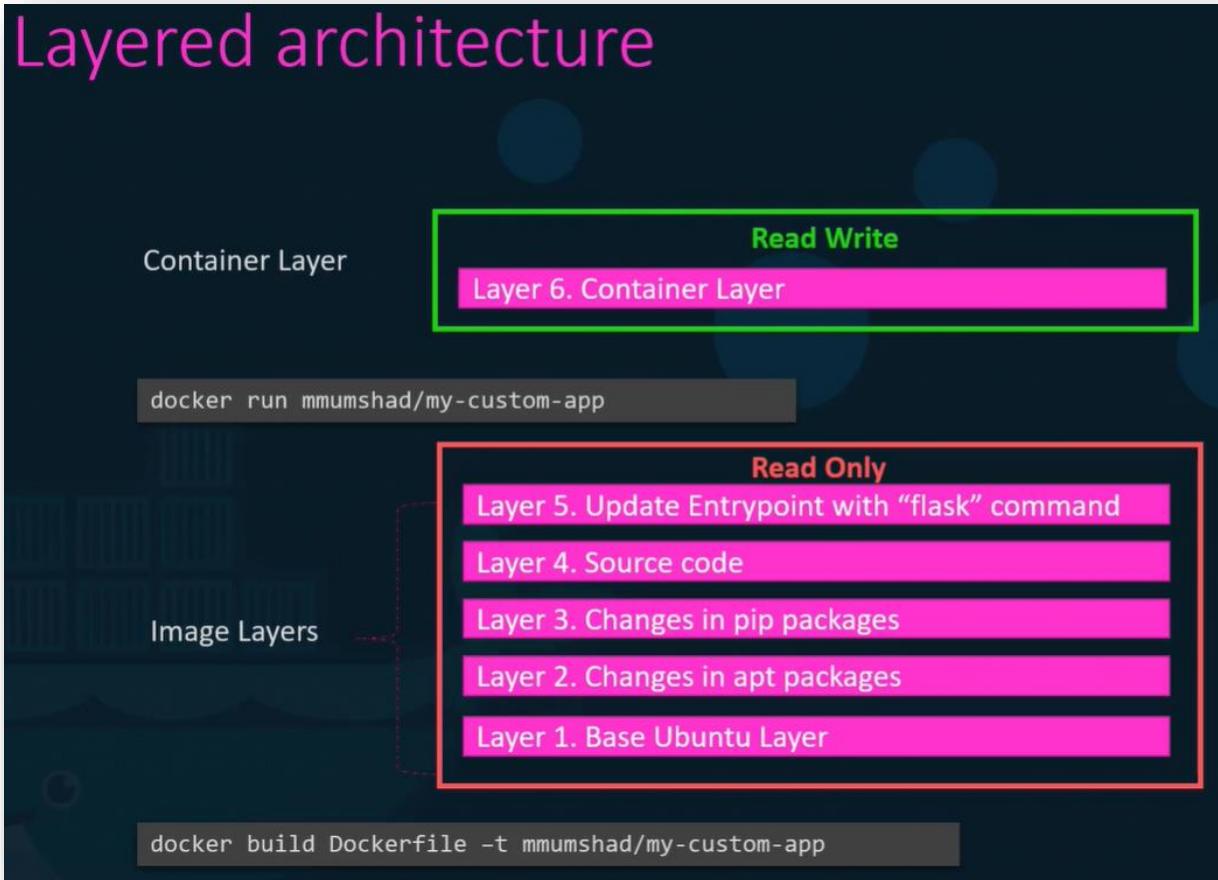


Step 2.

Layered architecture



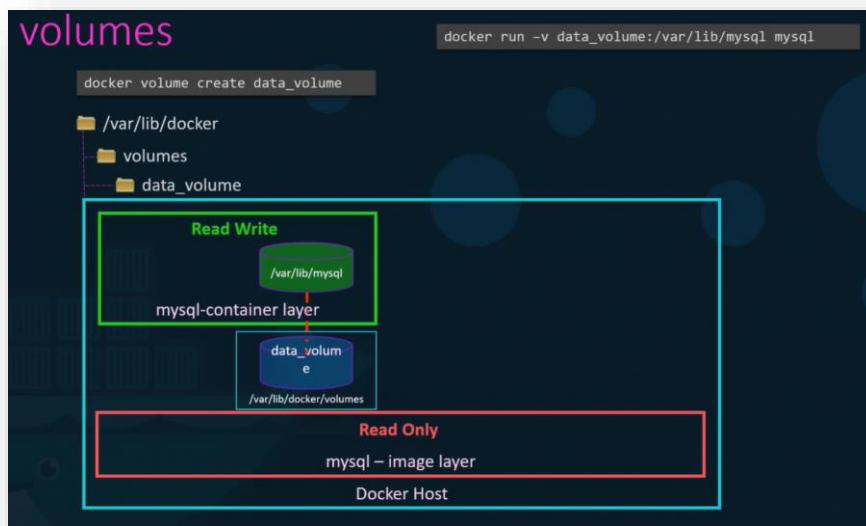
Step 3.



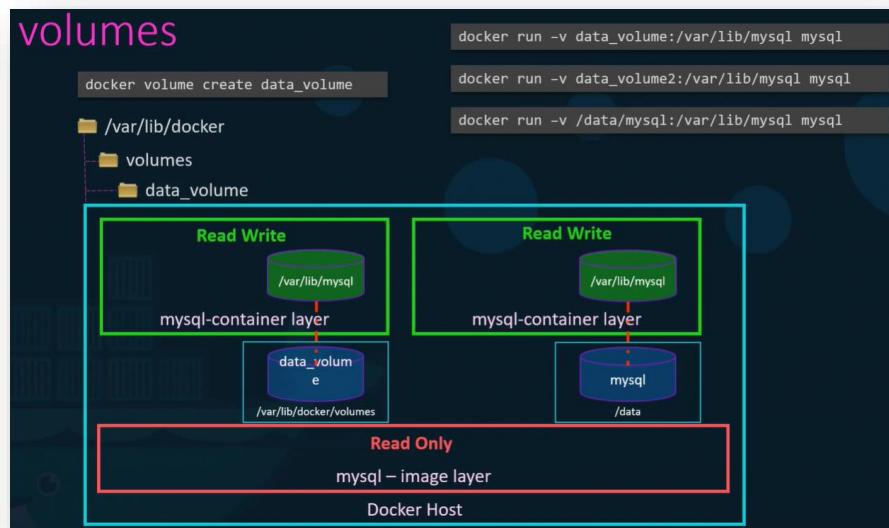
Step 4.



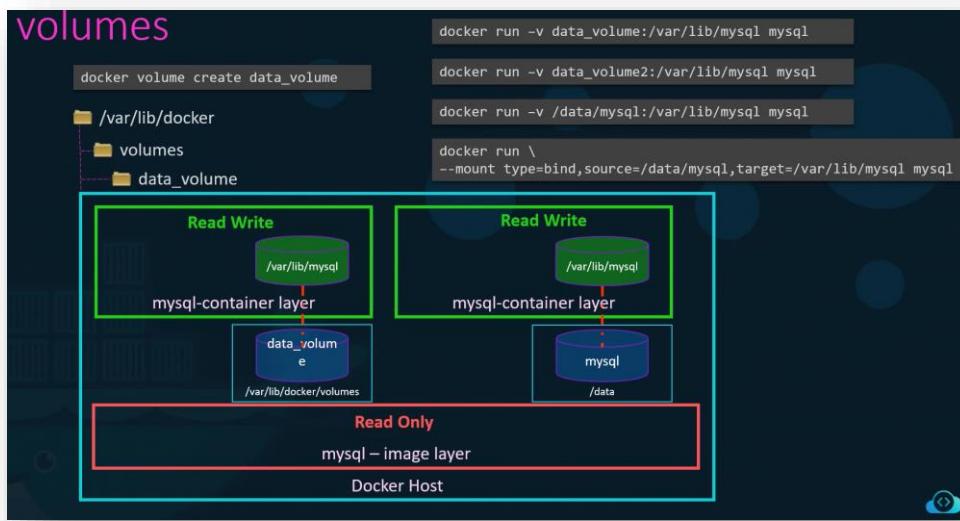
Step 5.



Step 6.



Step 7.



Step 8.

Storage drivers

- AUFS
- ZFS
- BTRFS
- Device Mapper
- Overlay
- Overlay2

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.

STORAGE DRIVERS

AUFS | ZFS | BTRFS | DEVICE MAPPER | OVERLAY

Step 2.

VOLUME DRIVERS

Local | Azure File Storage | Convoy |
DigitalOcean Block Storage | Flocker | gce-docker | GlusterFS
| NetApp | RexRay | Portworx | VMware vSphere Storage

Step 3.

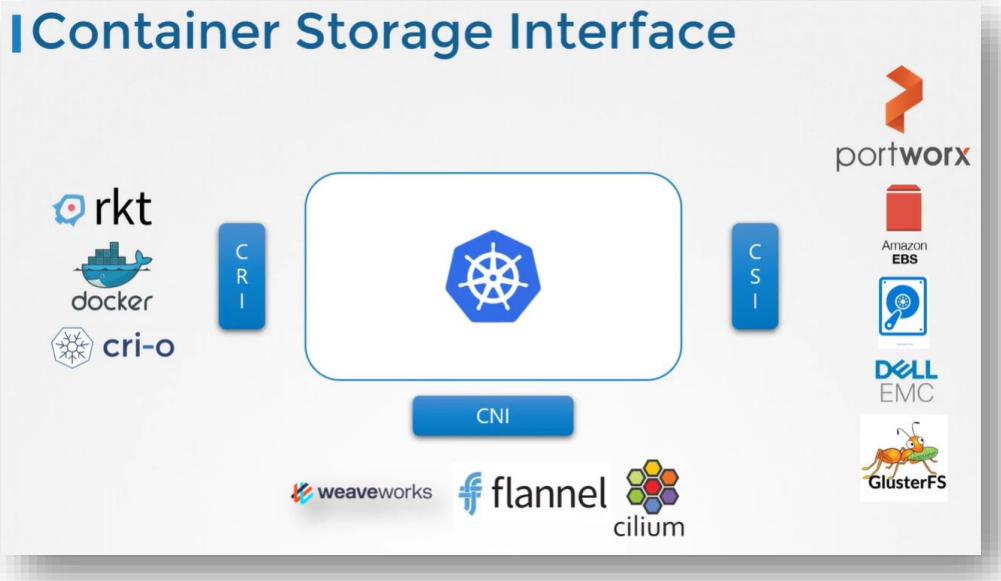
I VOLUME DRIVERS

```
▶ docker run -it \
  --name mysql
  --volume-driver rexray/ebs
  --mount src=ebs-vol,target=/var/lib/mysql
mysql
```



8.5 CONTAINER STORAGE INTERFACE

Step 1.

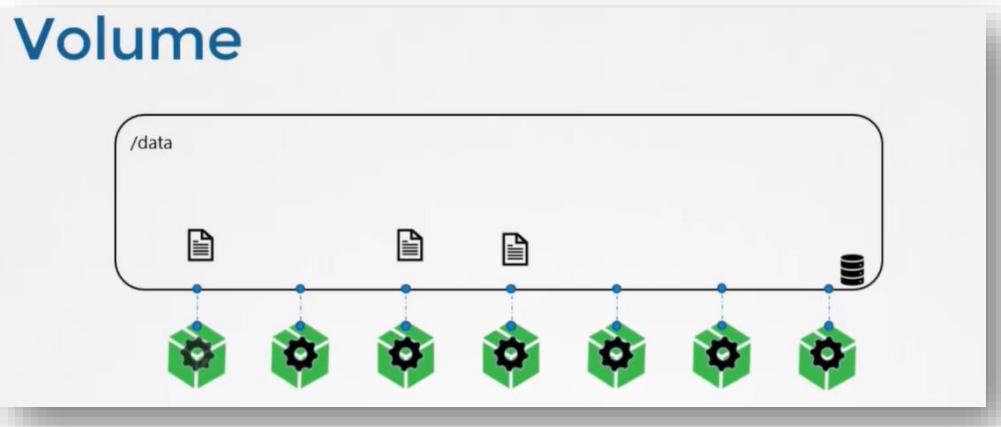


Step 2.



8.6 VOLUMES

Step 1.



Step 2.

```
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.

Volumes & Mounts

```
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
```

The diagram illustrates the relationship between a data volume and its mounting. On the left, a purple circle represents the data volume, containing icons for a file labeled '56' and a database. A blue arrow points from this volume to a white rectangular box representing a pod container. Inside the container, there is a green cube icon representing the /opt directory. On the right, a white box represents the host system, showing a folder icon labeled '/data' and a file icon labeled '56'. A blue arrow points from the host's /data directory to the pod's /opt directory, indicating the mapping.

Step 4.

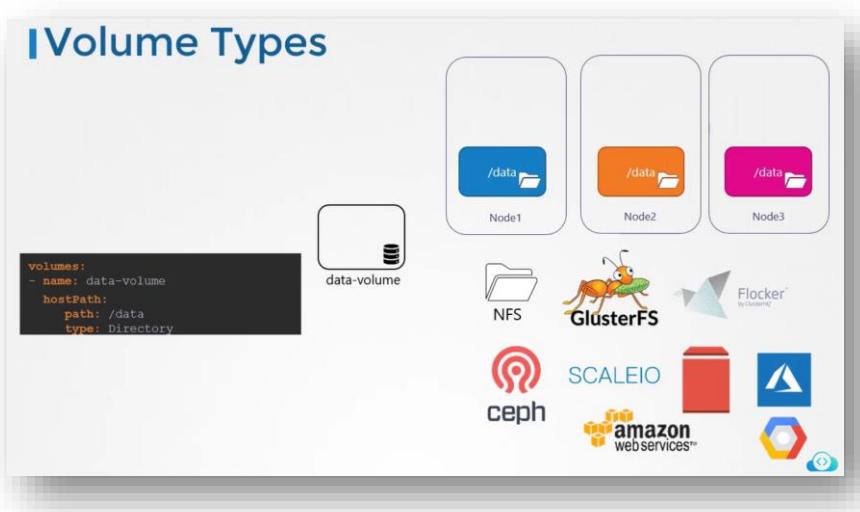
Volumes & Mounts

```
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

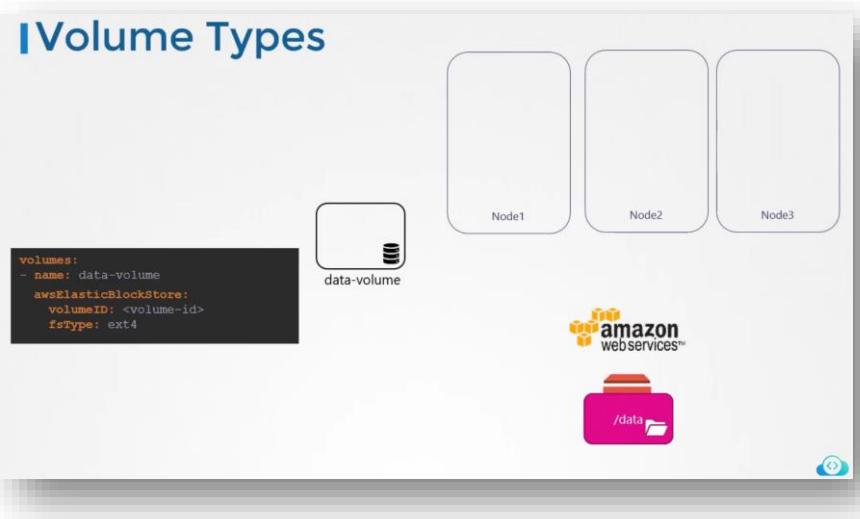
  volumes:
    - name: data-volume
      hostPath:
        path: /data
        type: Directory
```

This diagram shows the state of the system after Step 4. The data volume 'data-volume' is still present on the host system (Node1) at the path '/data', represented by a folder icon labeled '/data' and a file icon labeled '56'. However, the pod container no longer has a /opt directory or any associated mounts, as indicated by the empty white box representing the pod.

Step 5.

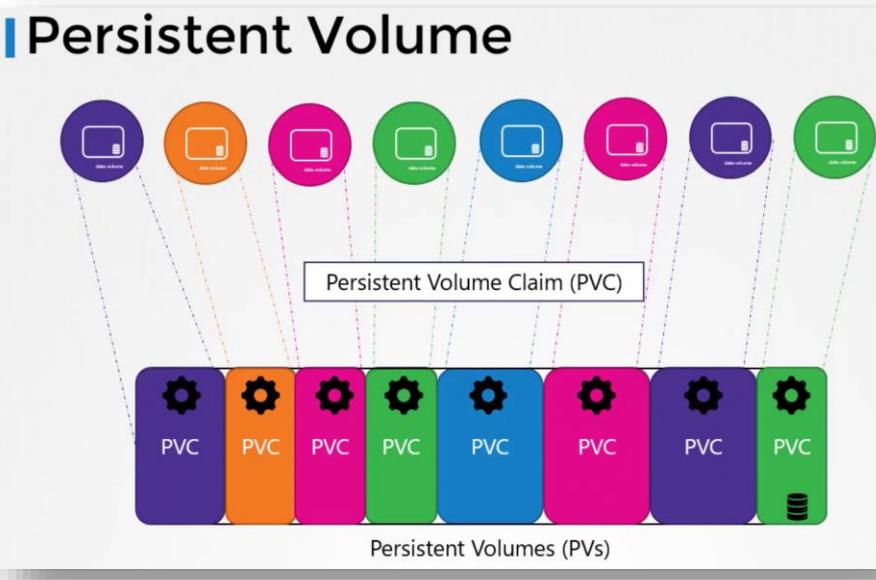


Step 6.



8.7 PERSISTANT VOLUMES

Step 1.



Step 2.

Persistent Volume

pv-definition.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-voll
spec:
  accessModes:
    - ReadWriteOnce
```

ReadOnlyMany

ReadWriteOnce

ReadWriteMany



Persistent Volume (PV)

Step 3.

Persistent Volume

pv-definition.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-voll
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  hostPath:
    path: /tmp/data
```



kubectl create -f pv-definition.yaml

Step 4. Local Volume

Persistent Volume

```
pv-definition.yaml
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-voll
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  awsElasticBlockStore:
    volumeID: <volume-id>
    fsType: ext4
```

```
▶ kubectl create -f pv-definition.yaml
```

```
▶ kubectl get persistentvolume
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
pv-voll	1Gi	RWO	Retain	Available				3m



Persistent Volume (PV)

Step 5. Cloud Storage

Persistent Volume

```
pv-definition.yaml
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-voll
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  hostPath:
    path: /tmp/data
```

```
▶ kubectl create -f pv-definition.yaml
```

```
▶ kubectl get persistentvolume
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
pv-voll	1Gi	RWO	Retain	Available				3m



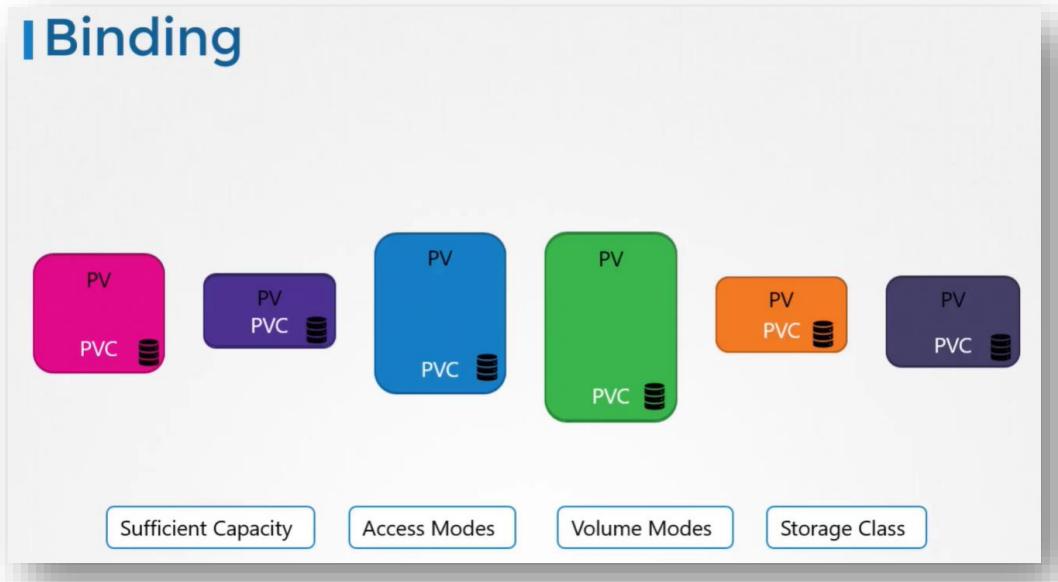
Persistent Volume (PV)

8.8 COMMANDLINE

- kubectl get pv
- kubectl describe pv pv_name

8.9 PERSISTANT VOLUME CLAIMS

Step 1.



Step 2.

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

Step 3.

Delete PVCs

```
kubectl delete persistentvolumeclaim myclaim  
persistentvolumeclaim "myclaim" deleted
```



```
persistentVolumeReclaimPolicy: Retain
```

Step 4.

Delete PVCs

```
kubectl delete persistentvolumeclaim myclaim  
persistentvolumeclaim "myclaim" deleted
```



```
persistentVolumeReclaimPolicy: Delete
```

Step 5.

Delete PVCs

```
kubectl delete persistentvolumeclaim myclaim  
persistentvolumeclaim "myclaim" deleted
```



```
persistentVolumeReclaimPolicy: Recycle
```

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

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

Step 4.

| Dynamic Provisioning

pv-definition.yaml

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

PV

SC

I 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
```

IDynamic 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
```

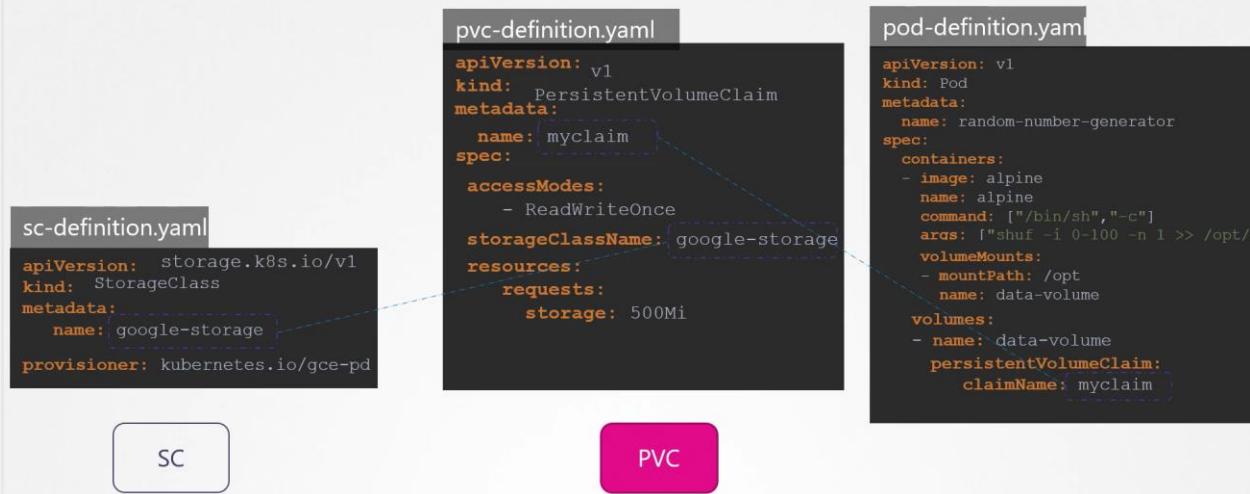
`pod-definition.yaml`

```
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/volume"]
      volumeMounts:
        - mountPath: /opt
          name: data-volume
      volumes:
        - name: data-volume
          persistentVolumeClaim:
            claimName: myclaim
```



Step 7.

| Dynamic Provisioning



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
```

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

I Storage Class

sc-definition.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard
  replication-type: none
```

sc-gold-definition.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
  replication-type: none
```

Silver
SC

Gold
SC

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 replicases (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. KUBERNETES 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 --dearmor -o
/etc/apt/trusted.gpg.d/docker.gpg

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu ${lsb_release -c} 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.egkshbevkiyngt \ --discovery-token-ca-cert-hash sha256:063117edb34c465fd0a6328312abb3199c60b19ecaf5374b792b0dd4709de1a1s
```

```
kubeadm join 10.251.0.11:6443 --token roatlj.wphhue848fh9oo40 \ --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 --dearmor -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.
