Core Concepts – 19%

**Pod**

|  |  |
| --- | --- |
| 1 | Kubectl run –dry-run –o yaml nginx --image=nginx --restart=Never -l wf=ov,hs=karthi --env=mami=lax --env=cnt=jp -port=80 -- /bin/sh -c "echo hi world" > pod.yaml |
| 2 | kubectl run --generator=run-pod/v1 nginx --image=nginx --port=8080 --command echo hi |
| 3 | kubectl get pod podname –o yaml –export > pod.yaml |
| 4 | kubectl set image pod/nginx nginx=nginx:1.7.1 |
| 5 | kubectl exec podname -it bash **or** kubectl exec podname –it -- /bin/sh # get inside the pod |

**ReplicationController**

|  |  |
| --- | --- |
| 1 | kubectl run replicontrolr --generator=run/v1 --image=redis --replicas=2 --dry-run -o yaml |

**ReplicaSet**

|  |  |
| --- | --- |
| 1 | kubectl run --generator=deployment/v1beta1 nginx --image=nginx --dry-run --replicas=4 -o yaml  // edit the Deployment to replicaSet , remove strategy and empty properties |
| 2 | kubectl scale –replicas=3 rc/rc1 rc/rc2 rc/rc3 | kubectl scale deploy mydeploy –replicas=5 |

**Deployments**

|  |  |
| --- | --- |
| 1 | kubectl run --dry-run nginx --image=nginx -l wf=ov,hs=karthi --env=mami=lax --env=cnt=jp --port=80 -o yaml --  replicas=5 -- /bin/sh -c "echo hi world" > dep.yaml |
| 2 | kubectl set image deploy nginx nginx=nginx:1.7.1 |
| 3 | kubectl run --generator=deployment/v1beta1 nginx --image=nginx --dry-run --replicas=4 -o yaml |
| 4 | kubectl autoscale deploy nginx --min=5 --max=10 --cpu-percent=80 --dry-run -o yaml |

**Service:**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | kubectl create service clusterip ngservice --tcp=80:80 --dry-run -o yaml | | |
| 2 | kubectl create service nodeport nginx --tcp=80:8000 --node-port=30080 --dry-run -o yaml | | |
| 3 | kubectl expose deployment nginx --type=NodePort --port=80 --target-port=8000 --name=nginx-serv --dry-run -o yaml kubectl expose deployment nginx –port=80 –target-port=8000 | | |
| 4 | kubectl run --image=nginx ng --port=8080 --expose --dry-run -o yaml | | |
|  | kubectl run nginx --image=nginx --restart=Never --port=80 --expose |  |

**NameSpaces:**

|  |  |
| --- | --- |
| 1. | Kubectl create namespace mynamespace |
| 2 | kubectl get all --all-namespaces |
| 3 | kubectl run nginx –image=nginx –n mynamespace |

Scheduling – 5%

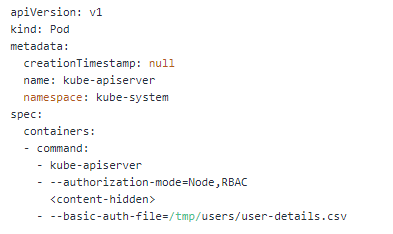
Security – 12 %

With Files

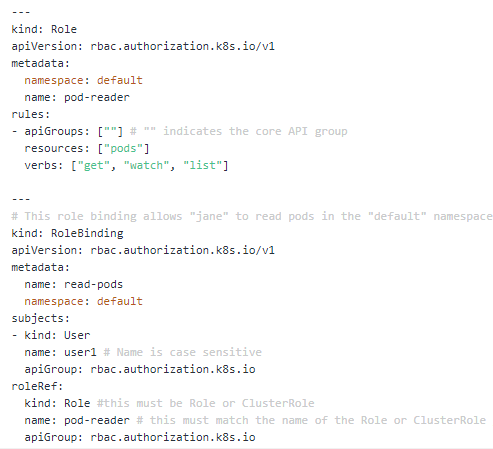
* Create a file with user details locally at /tmp/users/user-details.csv

1. password123,user1,u0001
2. password123,user2,u0002

* Modify the kube-apiserver startup options to include the basic-auth file /etc/kubernetes/manifests/kube-apiserver.yaml



* Create the necessary roles and role bindings for these users:



curl -v -k https://localhost:6443/api/v1/pods -u "user1:password123"

* Generating CA key & Certificate

# Create private key for CA

**openssl genrsa -out ca.key 2048**

# Create CSR using the private key

**openssl req -new -key ca.key -subj "/CN=KUBERNETES-CA" -out ca.csr**

# Self sign the csr using its own private key

**openssl x509 -req -in ca.csr -signkey ca.key -CAcreateserial -out ca.crt -days 1000**

* Generating Admin key & Certificate

# Geenrate private key for admin user

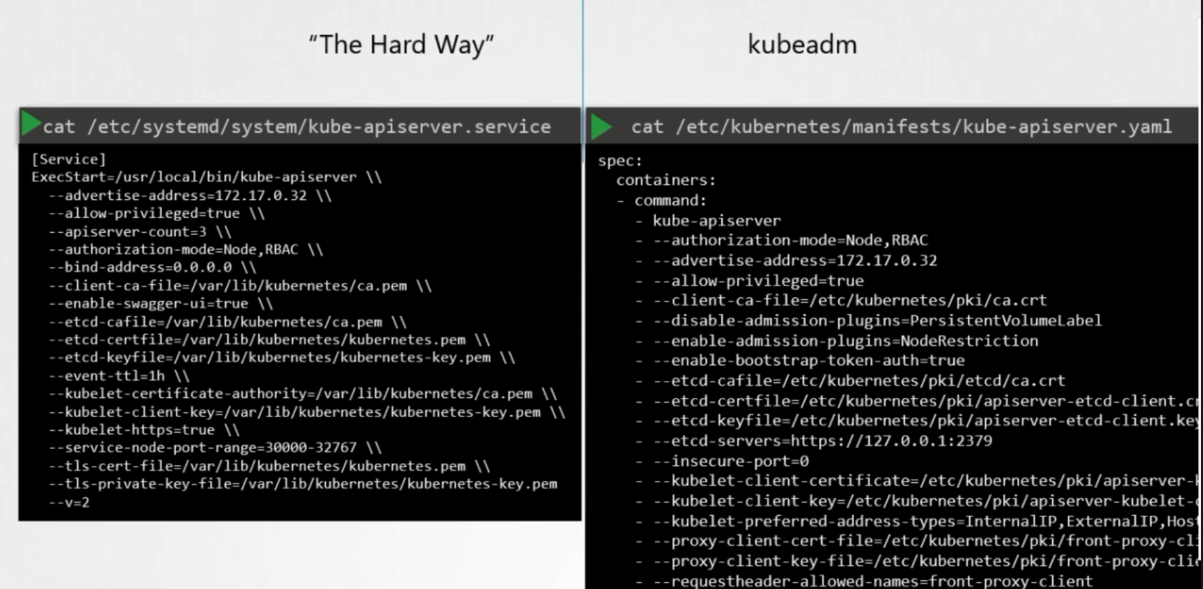
**openssl genrsa -out admin.key 2048**

# Generate CSR for admin user. Note the OU.

**openssl req -new -key admin.key -subj "/CN=admin/O=system:masters" -out admin.csr**

# Sign certificate for admin user using CA servers private key

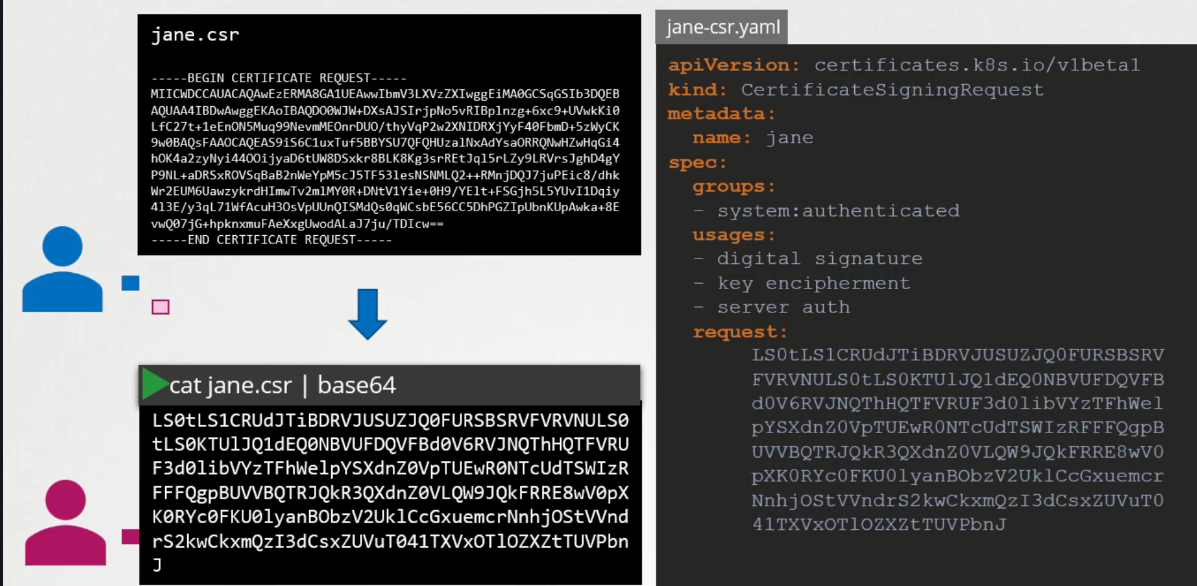
**openssl x509 -req -in admin.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out admin.crt -days 1000**

* Viewing certificate

**Openssl x509 –in /file-path.crt –text –noout**

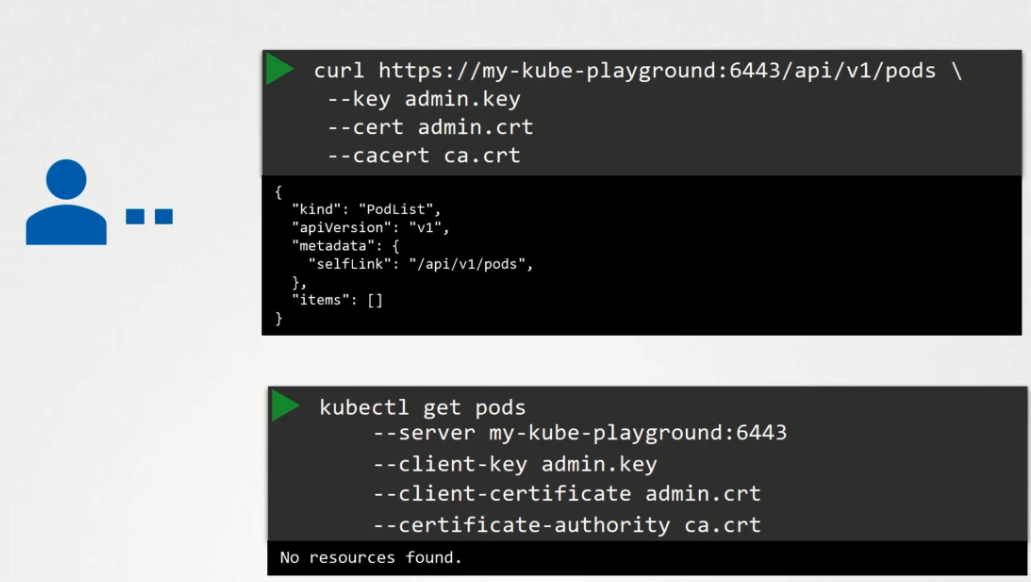
New user – Flow

1. Create the key : **openssl genrsa –out karthik.key 2049**
2. Create CsigningReq to Admin: **openssl req –new –key karthik.key –subj “/CN=karthik” –out karthik.csr**
3. Admin takes the key and create the certificate Signing Object **cat karthik.csr | base64**



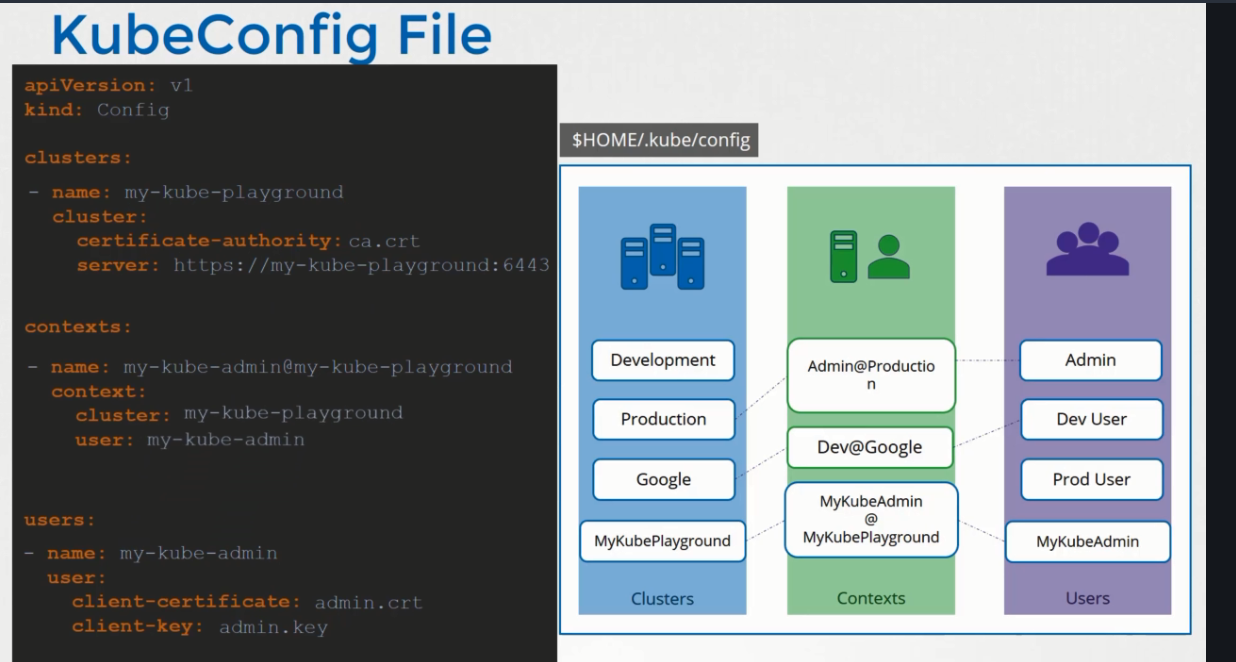
1. **Kubectl get csr**
2. **Kubectl certificate approve karthik**

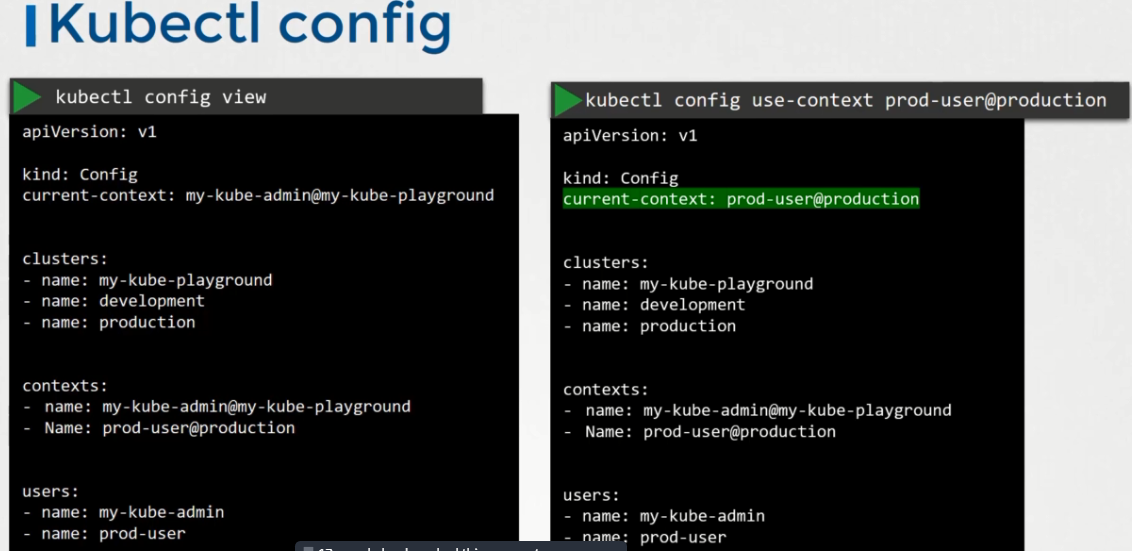
* **KubeConfig**

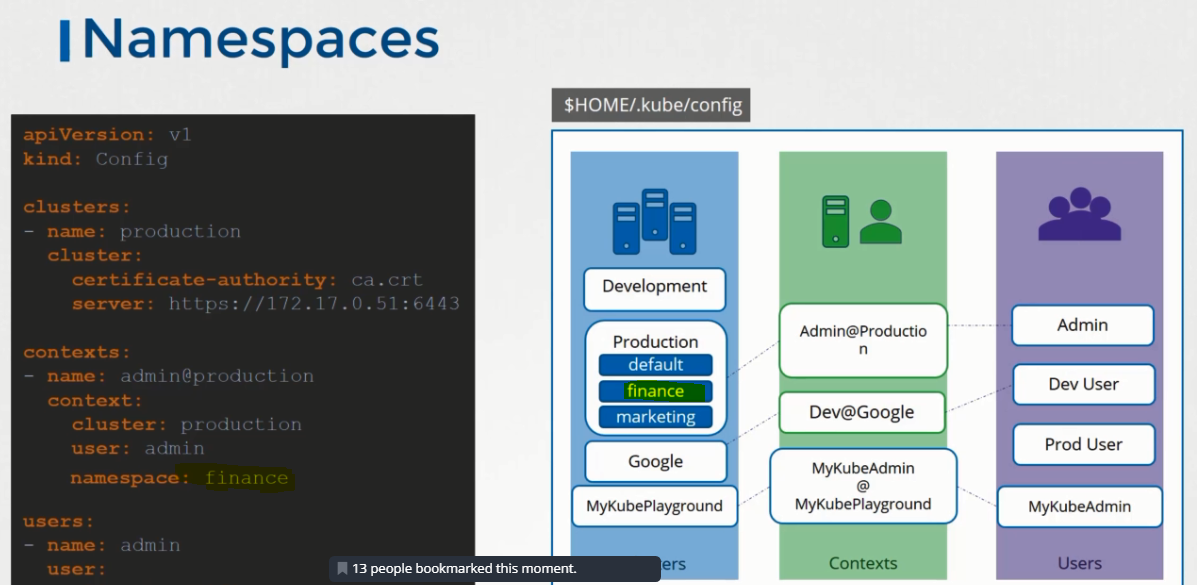
 Every time we cannot type the cert details in the command

Config file has 3 sections. 1. Cluster 2. Contexts 3.Users

Context is bridge the cluster and Users. Providing access to which user has to use which cluster





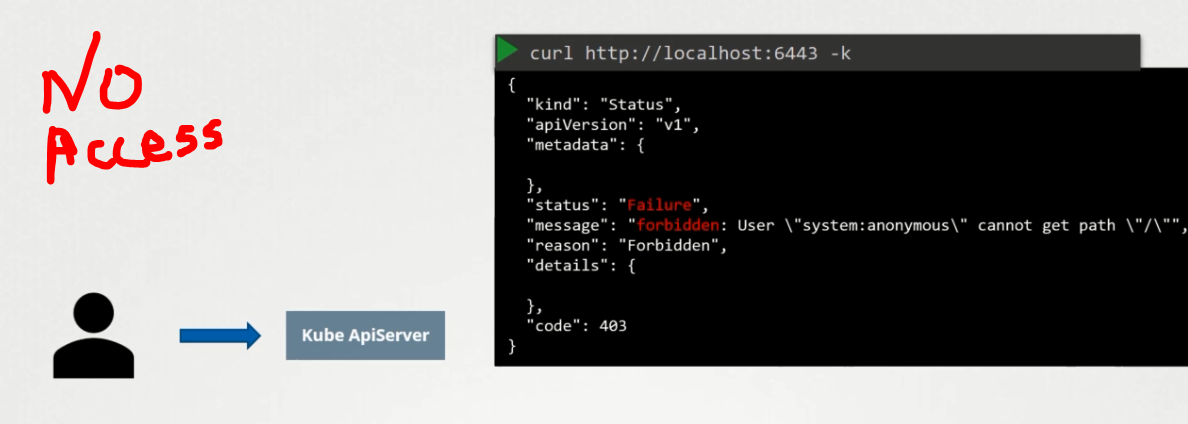


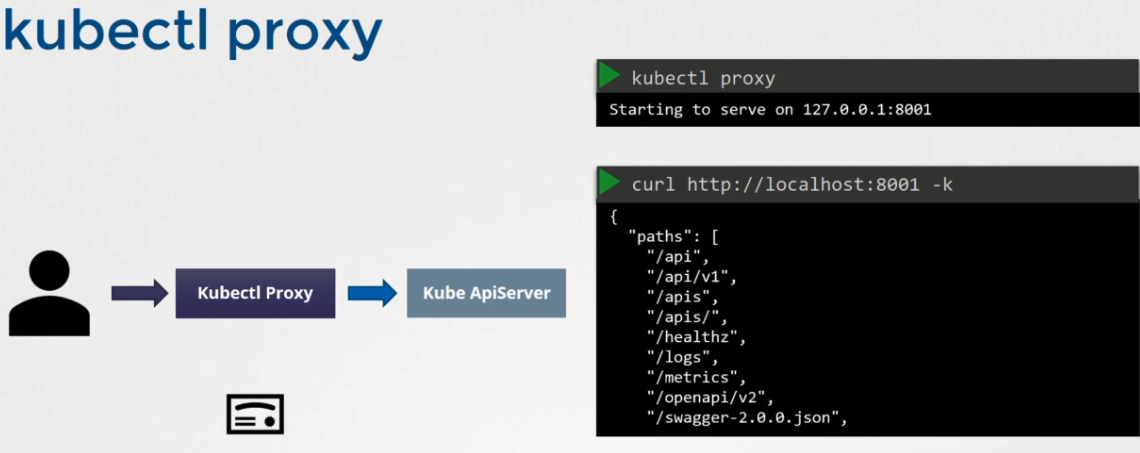
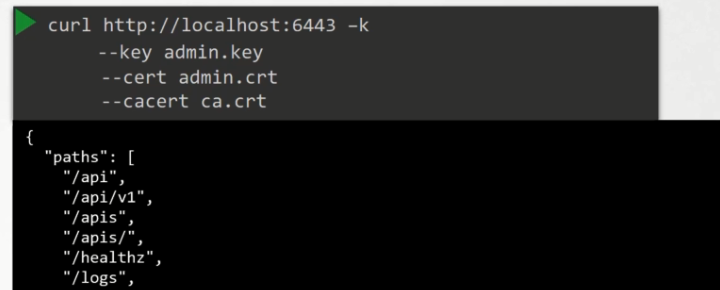


// different kubeconfig file

**kubectl config --kubeconfig=/root/my-kube-config use-context research**

* **Kubectl Proxy**

 bec certificate required



Kubectl – http proxy to access the kube api server; kube proxy – responsible to connect all the pods/services



kubectl create role dev --verb=create,list,delete --resource=pods --dry-run -o yaml

kubectl create rolebinding dev-bind-michelle --role=dev --user=michelle --dry-run -o yaml



Commands:

Kubectl get roles

Kubectl get rolebindings

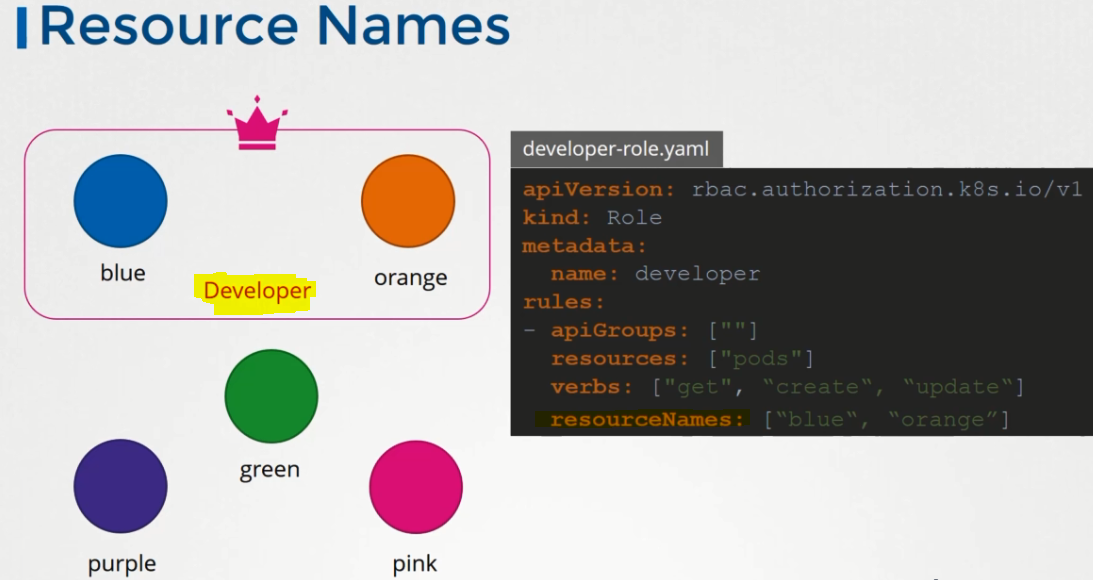
Kubectl describe role developer

Kubectl describe rolebinding devuser-developer-binding

Check Access:

Kubectl auth can-i create pods

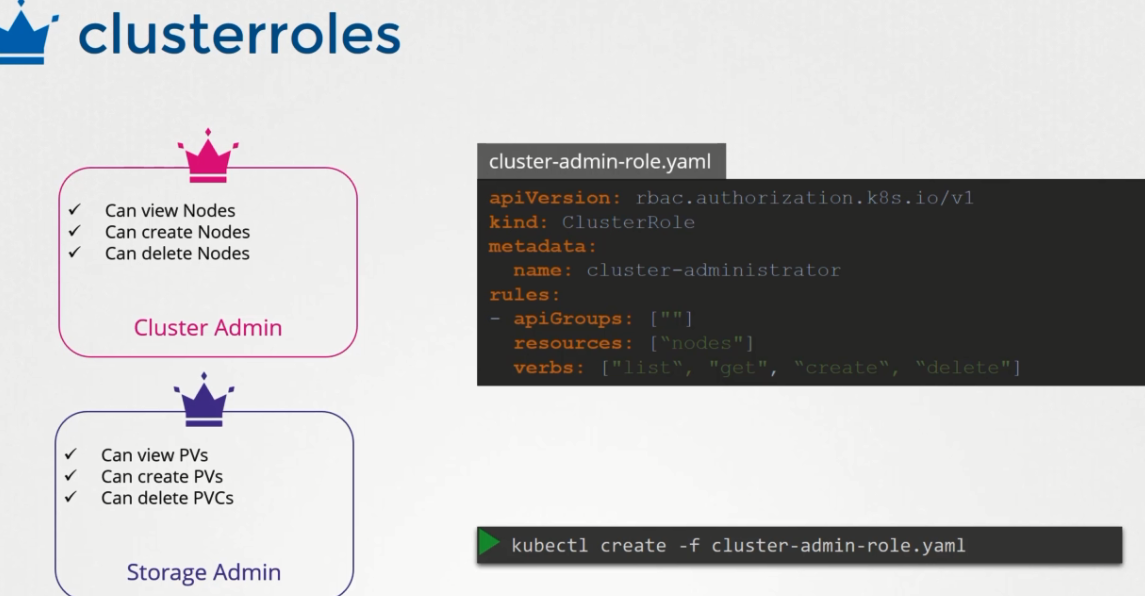
Kubectl auth can-I create deployments –as dev-user –namespace test

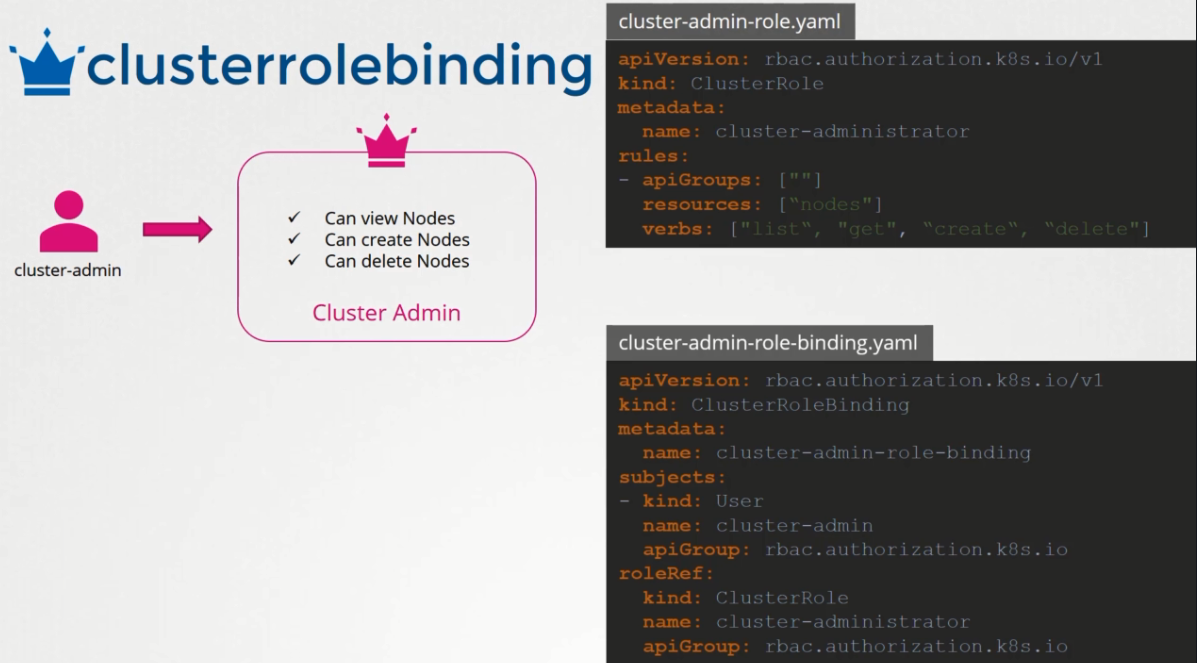
Pod-level restrictions 

Cluster-role and rolebinding

kubectl create clusterrole node-maintainer --verb=create,list,delete,watch --resource=node

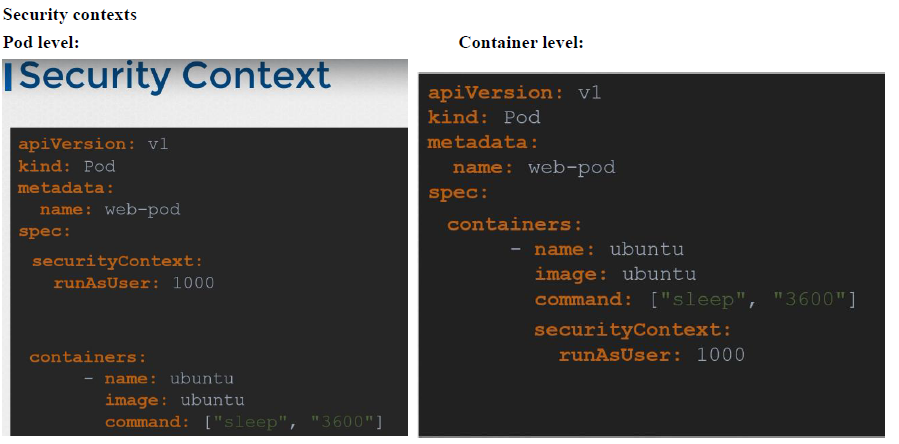
kubectl clusterrolebinding node-michelle --user=michelle --clusterrole=node-maintainer

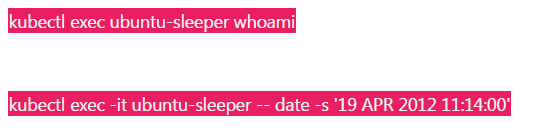


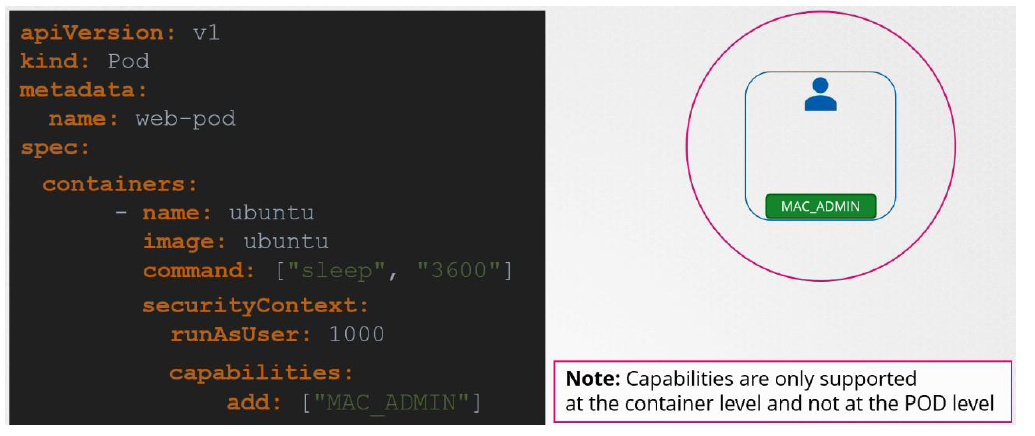
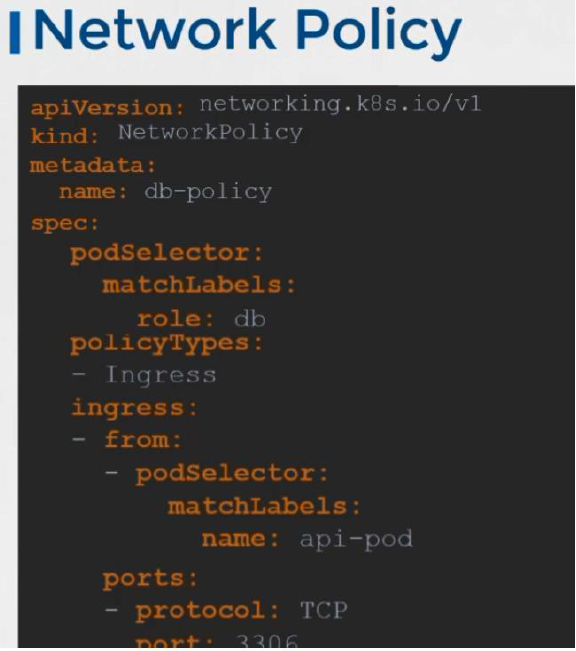


kubectl get clusterroles --no-headers | wc -l

**Security context**





K get netpol

**Cluster Maintanence: 11 %**

kubectl drain node01 --ignore-daemonsets

// pods ll be moved to other node, Node in Unschedulable status

kubectl cordon node01

// pod ll be in the same node, new pod cannot be scheduled

kubectl uncordon node01 // new pod can be scheduled

**Kubeadm upgrade**

Kubeadm upgrade plan

**Master Node**

* kubectl drain master --ignore-daemonsets
* apt install kubeadm=1.12.0-00
* kubeadm upgrade apply v1.12.0
* apt install kubelet=1.12.0-00
* Systemctl restart kubelet
* Kubectl get nodes

**Worker Node**

* Kubectl drain node1 --ignore-daemonsets
* 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
* Kubectl uncordon node1

**Backup**

Resource config : kubectl get all –all-namespaces –o yaml > all-resources.yaml

ETCDCTL\_API=3 etcdctl snapshot save snapshot.db

Ls to list the db file

ETCDCTL\_API=3 etcdctl snapshot status snapshot.db

To restore the backup

// during restore, we must specify new cluster token to avoid collision with another cluster. We also need to provide –endpoint,cacert,etcd cert, etcd key.

Service kube-apiserver stop

ETCDCTL\_API=3 etcdctl snapshot restore snapshot.db

--data-dir /path/of/backup/file

--initial-cluster master1­=https://ip:2380

--initial-cluster-token etcd-cluster

--initial-advertise-peer-urls https://${INTERNAL\_IP}:2380

Start etcd service with initial-cluster-token etcd-cluster , --data-dir /path/of/backup/file

Systemctl daemon-reload

Service etcd restart

Service kube-apiserver start