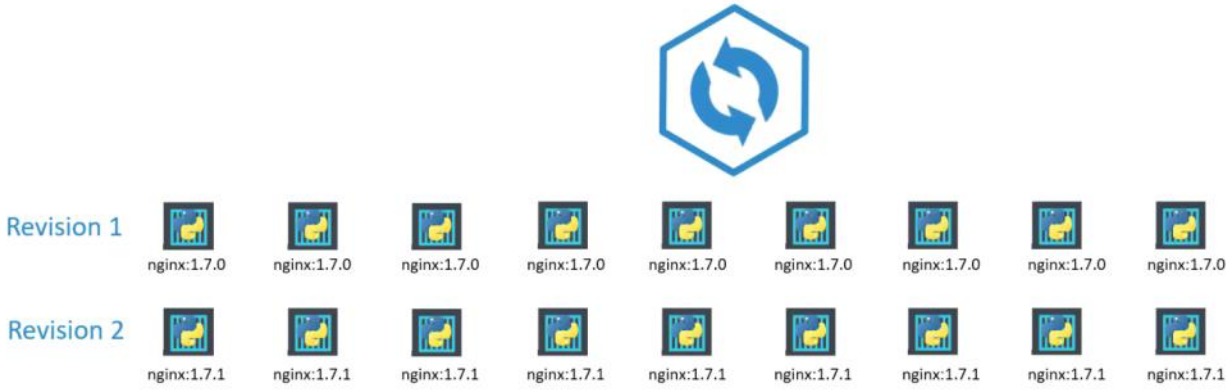
Application Lifecycle Management

Friday, June 14, 2024 9:23 AM

Rollout and Versioning



Rollout Command

```
> kubectl rollout status deployment/myapp-deployment
Waiting for rollout to finish: 0 of 10 updated replicas are available...
Waiting for rollout to finish: 1 of 10 updated replicas are available...
Waiting for rollout to finish: 2 of 10 updated replicas are available...
Waiting for rollout to finish: 3 of 10 updated replicas are available...
Waiting for rollout to finish: 4 of 10 updated replicas are available...
Waiting for rollout to finish: 5 of 10 updated replicas are available...
Waiting for rollout to finish: 6 of 10 updated replicas are available...
Waiting for rollout to finish: 7 of 10 updated replicas are available...
Waiting for rollout to finish: 8 of 10 updated replicas are available...
Waiting for rollout to finish: 9 of 10 updated replicas are available...
deployment "myapp-deployment" successfully rolled out
```

```
> kubectl rollout history deployment/myapp-deployment
deployments "myapp-deployment"
REVISION  CHANGE-CAUSE
1         <none>
2         kubectl apply --filename=deployment-definition.yml --record=true
```

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

```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
```

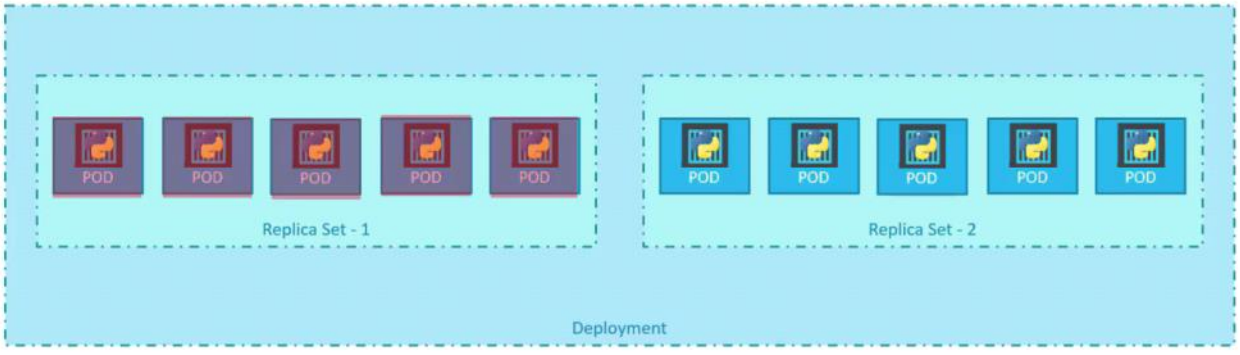
```
C:\Kubernetes>kubectl describe deployment myapp-deployment
Name:                   myapp-deployment
Namespace:              default
CreationTimestamp:      Sat, 03 Mar 2018 17:01:55 +0800
Labels:                 app=myapp
                        type=front-end
Annotations:            deployment.kubernetes.io/revision=2
                        kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"Deployment","m
s\\Google...
                        kubernetes.io/change-cause=kubectl apply --filename=d:\Mumshad Files\Google Drive\Udemy\Kubernet
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-54c7d6ccc (5/5 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  11m   deployment-controller  Scaled up replica set myapp-deployment-6795844b58 to 5
  Normal  ScalingReplicaSet  1m    deployment-controller  Scaled down replica set myapp-deployment-6795844b58 to 0
  Normal  ScalingReplicaSet  56s   deployment-controller  Scaled up replica set myapp-deployment-54c7d6ccc to 5
```

Recreate

```
C:\Kubernetes>kubectl describe deployment myapp-deployment
Name:                   myapp-deployment
Namespace:              default
CreationTimestamp:      Sat, 03 Mar 2018 17:16:53 +0800
Labels:                 app=myapp
                        type=front-end
Annotations:            deployment.kubernetes.io/revision=2
                        kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"Deployment","metadat
Files\\Google...
                        kubernetes.io/change-cause=kubectl apply --filename=d:\Mumshad Files\Google Drive\Udemy\Kubernetes\De
Selector:               type=front-end
Replicas:               5 desired | 5 updated | 6 total | 4 available | 2 unavailable
StrategyType:           RollingUpdate
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    ReplicaSetUpdated
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-7d57dbdb8d 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
```

RollingUpdate

Upgrades

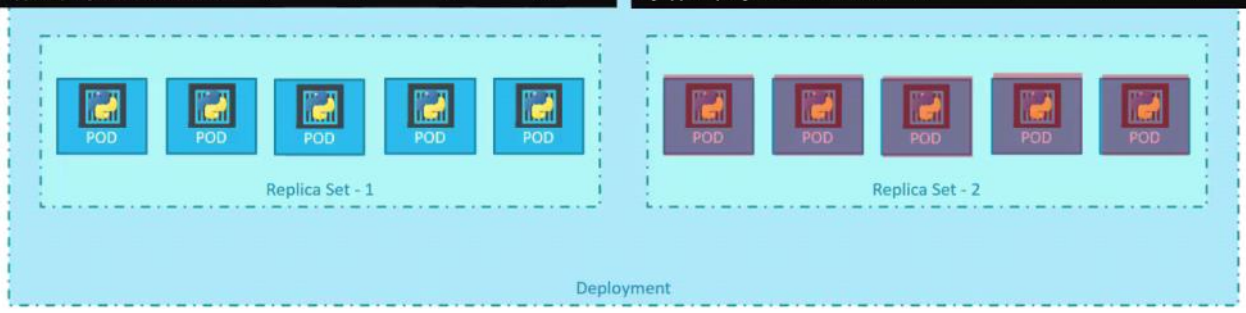



```
> kubectl get replicasets
NAME                          DESIRED   CURRENT   READY   AGE
myapp-deployment-67c749c58c   0         0         0       22m
myapp-deployment-7d57dbdb8d   5         5         5       20m
```

Rollback

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

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



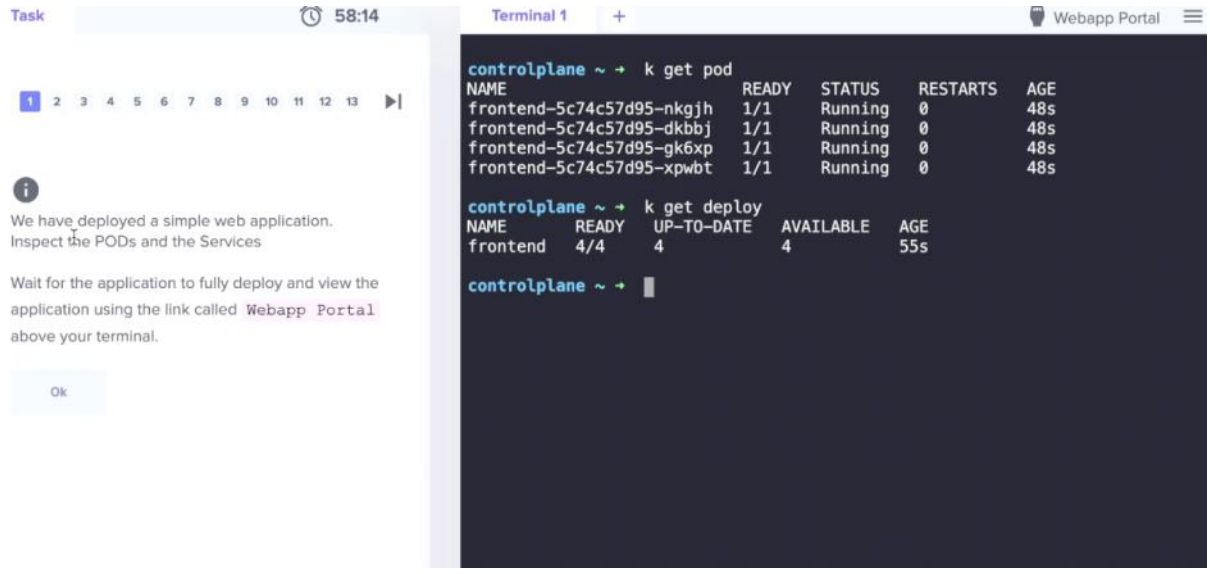

```
> kubectl rollout undo deployment/myapp-deployment
deployment "myapp-deployment" rolled back
```

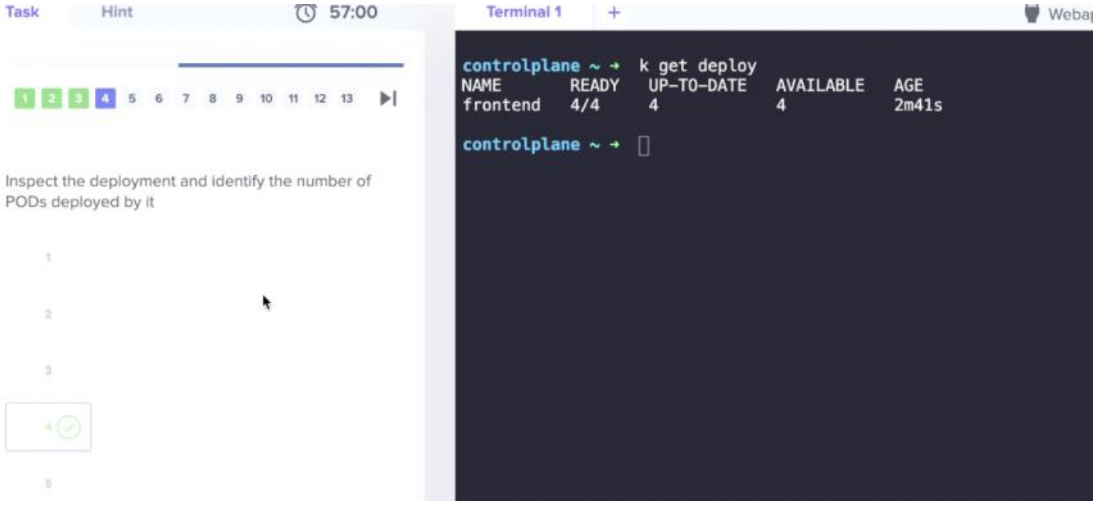
Summarize Commands

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

practice - rolling updates

Friday, June 14, 2024 7:17 PM





1 2 3 4 5 6 7 8 9 10 11 12 13

What container image is used to deploy the applications?

- kodekloud/webapp-color:v1
- kodekloud/webapp-color:v2
- kodekloud
- webapp
- simple-webapp

```
controlplane ~ ➜  k get deploy
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
frontend   4/4     4            4           2m41s

controlplane ~ ➜  k describe deploy frontend
Name:                   frontend
Namespace:              default
CreationTimestamp:      Sat, 16 Apr 2022 20:35:48 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               name=webapp
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        20
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  name=webapp
  Containers:
   simple-webapp:
    Image:        kodekloud/webapp-color:v1
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   frontend-5c74c57d95 (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  3m4s  deployment-controller  Scaled up replica set frontend-5c74c57d95 to 4
```

1 2 3 4 5 6 7 8 9 10 11 12 13

Inspect the deployment and identify the current strategy

- RollingUpdate
- ReCreate
- RollOut
- RollBack

```
controlplane ~ ➜  k get deploy
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
frontend   4/4     4            4           2m41s

controlplane ~ ➜  k describe deploy frontend
Name:                   frontend
Namespace:              default
CreationTimestamp:      Sat, 16 Apr 2022 20:35:48 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               name=webapp
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        20
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  name=webapp
  Containers:
   simple-webapp:
    Image:        kodekloud/webapp-color:v1
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   frontend-5c74c57d95 (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  3m4s  deployment-controller  Scaled up replica set frontend-5c74c57d95 to 4

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13

If you were to upgrade the application now what would happen?

All PODs are taken down before upgrading any

PODs are upgraded few at a time

```
controlplane ~ ➜  k get deploy
NAME       READY   UP-TO-DATE
frontend   4/4     4

controlplane ~ ➜  k describe de
Name:                   fronten
Namespace:              default
CreationTimestamp:      Sat, 16
Labels:                 <none>
Annotations:            deploym
Selector:               name=we
Replicas:               4 desir
StrategyType:           Rolling
MinReadySeconds:        20
RollingUpdateStrategy:  25% max
```

1 2 3 4 5 6 7 8 9 10 11 12 13

Let us try that. Upgrade the application by setting the image on the deployment to

`kodekloud/webapp-color:v2`

Do not delete and re-create the deployment. Only set the new image name for the existing deployment.

Check

- Deployment Name: frontend
- Deployment Image: kodekloud/webapp-color:v2

```
controlplane ~ ➜

controlplane ~ ➜  k set image deploy frontend ^C

controlplane ~ ✖ k describe deploy frontend
Name:                   frontend
Namespace:              default
CreationTimestamp:      Sat, 16 Apr 2022 20:35:48 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               name=webapp
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        20
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  name=webapp
  Containers:
   simple-webapp:
    Image:        kodekloud/webapp-color:v1
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   frontend-5c74c57d95 (4/4 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  5m14s  deployment-controller  Scaled up replica set frontend-5c74c57d95 to
4
```

```
controlplane ~ ➜  k set image deploy frontend simple-webapp=kodekloud/webapp-color:v2
deployment.apps/frontend image updated

controlplane ~ ➜

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13

Run the script `curl-test.sh` again. Notice the requests now hit both the old and newer versions. However none of them fail.

Execute the script at `/root/curl-test.sh`.

Ok

```
4
  Normal  ScalingReplicaSet  6s     deployment-controller  Scaled up replica set frontend-d495f944b to 1
  Normal  ScalingReplicaSet  6s     deployment-controller  Scaled down replica set frontend-5c74c57d95 t
o 3                To exit full screen, press Esc
  Normal  ScalingReplicaSet  5s     deployment-controller  Scaled up replica set frontend-d495f944b to 2

controlplane ~ ➜  ./curl-test.sh
Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v2 ; Color: green OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK

Hello, Application Version: v1 ; Color: blue OK
```

1 2 3 4 5 6 7 8 9 10 11 12 13

Up to how many PODs can be down for upgrade at a time

Consider the current strategy settings and number of PODs - 4

1

2

3

4

```
controlplane ~ ➜  k set image deploy frontend simple-webapp=kodekloud/webapp-color:v2
deployment.apps/frontend image updated
                To exit full screen, press Esc
controlplane ~ ➜

controlplane ~ ➜  k describe deploy frontend
Name:                   frontend
Namespace:              default
CreationTimestamp:      Sat, 16 Apr 2022 20:35:48 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 2
Selector:               name=webapp
Replicas:               4 desired | 2 updated | 5 total | 3 available | 2 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        20
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  name=webapp
  Containers:
   simple-webapp:
    Image:        kodekloud/webapp-color:v2
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    ReplicaSetUpdated
OldReplicaSets:  frontend-5c74c57d95 (3/3 replicas created)
NewReplicaSet:   frontend-d495f944b (2/2 replicas created)
```

here desired is 4 and rolling upadte strategy is 25% max which means that it will bring down max 25% at a time. hence 25% of 4 is 1.

1 2 3 4 5 6 7 8 9 10 11 12 13

Change the deployment strategy to `Recreate`

Delete and re-create the deployment if necessary. Only update the strategy type for the existing deployment.

Check

- Deployment Name: frontend
- Deployment Image: kodekloud/webapp-color:v2
- Strategy: Recreate

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "2"
  creationTimestamp: "2022-04-16T20:35:48Z"
  generation: 2
  name: frontend
  namespace: default
  resourceVersion: "1053"
  uid: a8b0a6e3-1c54-47e6-9e4a-b037be079fab
spec:
  minReadySeconds: 20
  progressDeadlineSeconds: 600
  replicas: 4
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      name: webapp
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: Recreate
  template:
    metadata:
      creationTimestamp: null
      labels:
        name: webapp
    spec:
      containers:
      - image: kodekloud/webapp-color:v2
        imagePullPolicy: IfNotPresent
        name: simple-webapp
        ports:
```

```
deployment.apps/frontend edited

controlplane ~ ➜  k describe deploy frontend
Name:                   frontend
Namespace:              default
CreationTimestamp:      Sat, 16 Apr 2022 20:35:48 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 2
Selector:               name=webapp
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
StrategyType:           Recreate
MinReadySeconds:        20
Pod Template:
  Labels:  name=webapp
  Containers:
   simple-webapp:
    Image:        kodekloud/webapp-color:v2
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
```

1 2 3 4 5 6 7 8 9 10 11 12 13

Upgrade the application by setting the image on the deployment to `kodekloud/webapp-color:v3`

Do not delete and re-create the deployment. Only set the new image name for the existing deployment.

Complete Next

- Deployment Name: frontend
- Deployment Image: kodekloud/webapp-color:v3

```
controlplane ~ ➜  k set image deploy frontend simple-webapp=kodekloud/webapp-color:v3
deployment.apps/frontend image updated

controlplane ~ ➜
```

Commands and Arguments- practice

Tuesday, June 18, 2024 12:30 PM

```
docker run --name ubuntu-sleeper ubuntu-sleeper
```

```
docker run --name ubuntu-sleeper ubuntu-sleeper 10
```

pod-definition.yml

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

FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]

pod-definition.yml

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

These are the commands which will run when the container starts.

1 2 3 4 5 6 7 8 9 10

How many PODs exist on the system?
in the current(default) namespace

0

...

2

3

4

5

```
controlplane ~ ➜  k get pdos
error: the server doesn't have a resource type "pdos"

controlplane ~ ✖ k get pods
NAME             READY   STATUS    RESTARTS   AGE
ubuntu-sleeper   1/1     Running   0          73s

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10

What is the command used to run the pod
`ubuntu-sleeper` ?

script.sh

No Command

sleep

sleep 3600

sleep 4800

```
controlplane ~ ➜  k describe pod ubuntu-sleeper
Name:         ubuntu-sleeper
Namespace:    default
Priority:     0
Node:         controlplane/172.25.0.64
Start Time:   Sat, 16 Apr 2022 22:13:09 +0000
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.42.0.9
IPs:
  IP:  10.42.0.9
Containers:
  ubuntu:
    Container ID:  containerd://6c27506bb09e96bbca115f2611b9cc93d26f9d85c9365543e8b6cdc1c1e6b82f
    Image:         ubuntu
    Image ID:      docker.io/library/ubuntu@sha256:9101220a875cee98b016668342c489ff0674f247f6ca20dfc91b9
1c0f28581ae
    Port:          <none>
    Host Port:     <none>
    Command:
      sleep
      4800
    State:          Running
      Started:      Sat, 16 Apr 2022 22:13:15 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-6w4dn (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
```

1 2 3 4 5 6 7 8 9 10

Create a pod with the ubuntu image to run a container to sleep for 5000 seconds. Modify the file `ubuntu-sleeper-2.yaml` .

Note: Only make the necessary changes. Do not modify the name.

Check

- Pod Name: ubuntu-sleeper-2
- Command: sleep 5000

```
controlplane ~ ➜  cat ubuntu-sleeper-2.yaml
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-sleeper-2
spec:
  containers:
  - name: ubuntu
    image: ubuntu

controlplane ~ ➜  vi ubuntu-sleeper-2.yaml

controlplane ~ ➜  cat ubuntu-sleeper-2.yaml
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-sleeper-2
spec:
  containers:
  - name: ubuntu
    image: ubuntu
    command: [ "sleep" ]
    args: [ "5000" ]

controlplane ~ ➜  kubectl create -f ubuntu-sleeper-2.yaml
```

1 2 3 4 5 6 7 8 9 10

Create a pod with the ubuntu image to run a container to sleep for 5000 seconds. Modify the file `ubuntu-sleeper-2.yaml`.

Note: Only make the necessary changes. Do not modify the name.

Check

- Pod Name: ubuntu-sleeper-2
- Command: sleep 5000

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

1 2 3 4 5 6 7 8 9 10

Create a pod using the file named `ubuntu-sleeper-3.yaml`. There is something wrong with it. Try to fix it!

Note: Only make the necessary changes. Do not modify the name.

Check

- Pod Name: ubuntu-sleeper-3
- Command: sleep 1200

```
controlplane ~ ➜  cat ubuntu-sleeper-3.yaml
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-sleeper-3
spec:
  containers:
  - name: ubuntu
    image: ubuntu
    command:
      - "sleep"
      - 1200

controlplane ~ ➜

controlplane ~ ➜  kubectl create -f ubuntu-sleeper-3.yaml
Error from server (BadRequest): error when creating "ubuntu-sleeper-3.yaml": Pod in version "v1" cannot
be handled as a Pod: json: cannot unmarshal number into Go struct field Container.spec.containers.comman
d of type string

controlplane ~ ✖ vi ubuntu-sleeper-3.yaml
```

1 2 3 4 5 6 7 8 9 10

Update pod `ubuntu-sleeper-3` to sleep for 2000 seconds.

Note: Only make the necessary changes. Do not modify the name of the pod. Delete and recreate the pod if necessary.

Check

- Pod Name: ubuntu-sleeper-3
- Command: sleep 2000

```
controlplane ~ ➜  kubectl edit pod ubuntu-sleeper
```

1 2 3 4 5 6 7 8 9 10

Update pod `ubuntu-sleeper-3` to sleep for 2000 seconds.
Note: Only make the necessary changes. Do not modify the name of the pod. Delete and recreate the pod if necessary.

Check

- Pod Name: ubuntu-sleeper-3
- Command: sleep 2000

```
  - lastProbeTime: null
    lastTransitionTime: "2022-04-16T22:20:22Z"
    status: "True"
    type: Initialized
  - lastProbeTime: null
    lastTransitionTime: "2022-04-16T22:20:25Z"
    status: "True"
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2022-04-16T22:20:25Z"
    status: "True"
    type: ContainersReady
  - lastProbeTime: null
    lastTransitionTime: "2022-04-16T22:20:22Z"
    status: "True"
    type: PodScheduled
  containerStatuses:
  - containerID: containerd://913c5de2b41512971009596a56ce2cc84d69338bacaaf7d2c245da030dc2a362
    image: docker.io/library/ubuntu:latest
    imageID: docker.io/library/ubuntu@sha256:9101220a875cee98b016668342c489ff0674f247f6ca20dfc91b91c0f28
581ae
    lastState: {}
    name: ubuntu
    ready: true
    restartCount: 0
    started: true
    state:
      running:
        startedAt: "2022-04-16T22:20:24Z"
  hostIP: 172.25.0.64
  phase: Running
  podIP: 10.42.0.11
  podIPs:
  - ip: 10.42.0.11
  qosClass: BestEffort
  startTime: "2022-04-16T22:20:22Z"

controlplane ~ ➜  kubectl replace --force -f /tmp/kubectl-edit-2693604347.yaml
pod "ubuntu-sleeper-3" deleted
```

1 2 3 4 5 6 7 8 9 10

Inspect the file `Dockerfile` given at `/root/webapp-color` directory. What command is run at container startup?

- python app.py
- Flask run
- nginx
- mysqld
- sleep 4800

```
controlplane ~ ➜  cat /root/webapp-color/Dockerfile
FROM python:3.6-alpine

RUN pip install flask

COPY . /opt/

EXPOSE 8080

WORKDIR /opt

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

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10

Inspect the file `Dockerfile2` given at `/root/webapp-color` directory. What command is run at container startup?

- python app.py
- python app.py --color red
- python app.py --color blue

```
controlplane ~ ➜  cat /root/webapp-color/Dockerfile
FROM python:3.6-alpine

RUN pip install flask

COPY . /opt/

EXPOSE 8080

WORKDIR /opt

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

controlplane ~ ➜  cat /root/webapp-color/Dockerfile2
FROM python:3.6-alpine

RUN pip install flask

COPY . /opt/

EXPOSE 8080

WORKDIR /opt

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

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

controlplane ~ ➜
```

To exit full screen, press Esc

1 2 3 4 5 6 7 8 9 10

Inspect the two files under directory `webapp-color-2`. What command is run at container startup?
Assume the image was created from the Dockerfile in this folder.

- python app.py
- python app.py --color red
- python app.py --color green
- --color green

```
controlplane ~ ➜  ls webapp-color-2
Dockerfile2            webapp-color-pod.yaml

controlplane ~ ➜  cat webapp-color-2/Dockerfile2
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-2/webapp-color-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: webapp-green
  labels:
      name: webapp-green
spec:
  containers:
  - name: simple-webapp
    image: kodekloud/webapp-color
    command: ["--color","green"]

controlplane ~ ➜
```

here cmd over rides entrypoint and arguments over ride command

1 2 3 4 5 6 7 8 9 10

Inspect the two files under directory `webapp-color-3`. What command is run at container startup?
Assume the image was created from the Dockerfile in this folder.

- python app.py
- python app.py --color pink
- python app.py --color green
- --color pink

```
controlplane ~ ➜  ls webapp-color-3/
Dockerfile2              webapp-color-pod-2.yaml

controlplane ~ ➜  cat webapp-color-3/Dockerfile2
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"]

controlplane ~ ➜
```

here cmd over rides entrypoint and arguments over ride command

1 2 3 4 5 6 7 8 9 10

Create a pod with the given specifications. By default it displays a `blue` background. Set the given command line arguments to change it to `green`

Check

- Pod Name: webapp-green
- Image: kodekloud/webapp-color
- Command line arguments: --color=green

```
controlplane ~ ➜  kubectl run webapp-green --image=kodekloud/webapp-color --dryn-run=client -o yaml
Error: unknown flag: --dryn-run
See 'kubectl run --help' for usage.
unknown flag: --dryn-run
See 'kubectl run --help' for usage.

controlplane ~ ✖ kubectl run webapp-green --image=kodekloud/webapp-color --dry-run=client -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: webapp-green
  name: webapp-green
spec:
  containers:
  - image: kodekloud/webapp-color
    name: webapp-green
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10

Create a pod with the given specifications. By default it displays a `blue` background. Set the given command line arguments to change it to `green`

Check

- Pod Name: webapp-green
- Image: kodekloud/webapp-color
- Command line arguments: --color=green

```
controlplane ~ ➜  ./webapp --color green^C

controlplane ~ ✖ kubectl run webapp <...> --color green ^C

controlplane ~ ✖ ^C

controlplane ~ ✖ kubectl run --help
```

1 2 3 4 5 6 7 8 9 10

Create a pod with the given specifications. By default it displays a `blue` background. Set the given command line arguments to change it to `green`

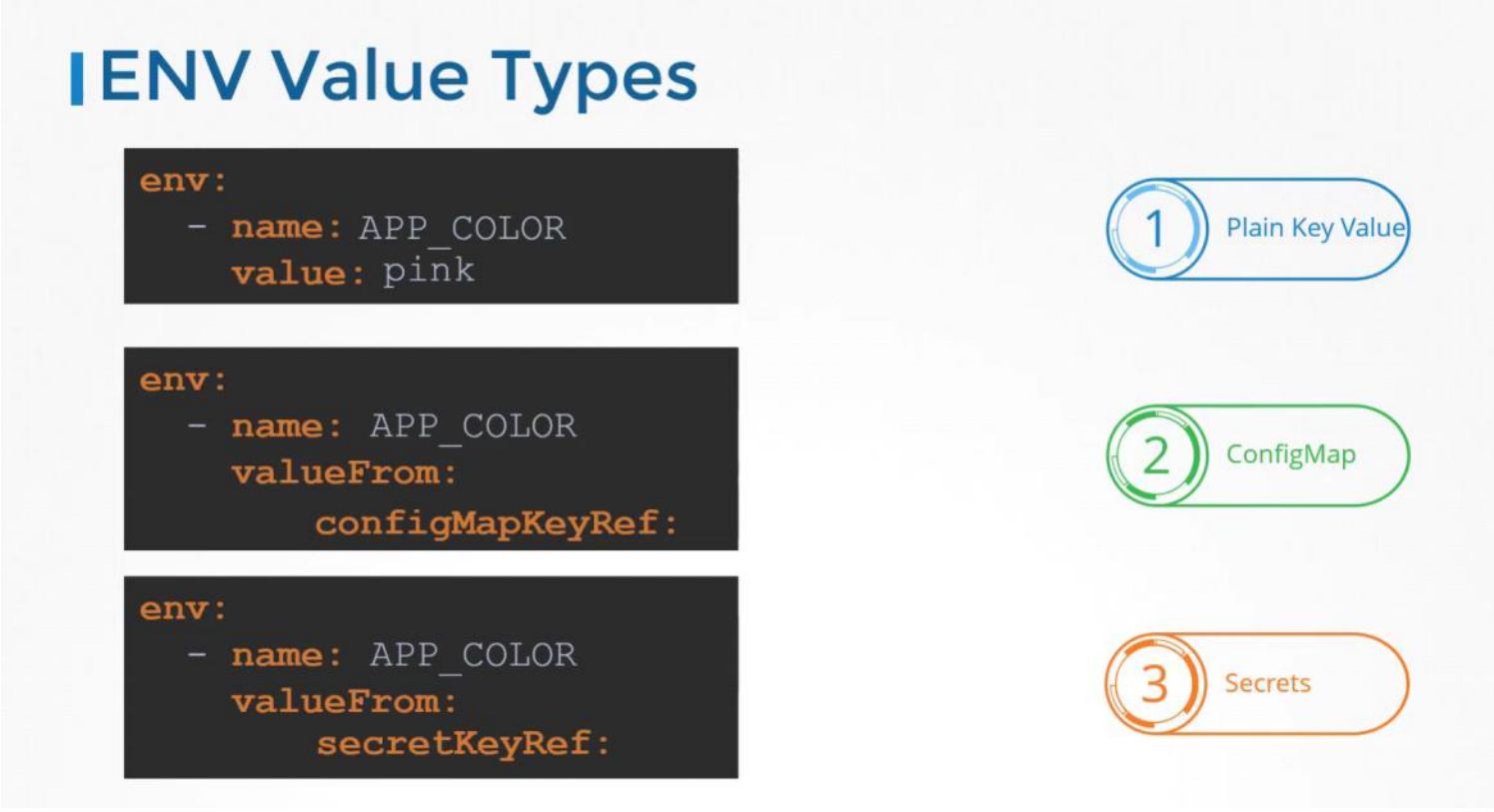
Check

- Pod Name: webapp-green
- Image: kodekloud/webapp-color
- Command line arguments: --color=green

```
controlplane ~ ➜  kubectl run webapp-green --image=kodekloud/webapp-color  -- --color green
pod/webapp-green created

controlplane ~ ➜  kubectl describe
```

Env variable in kubernetes

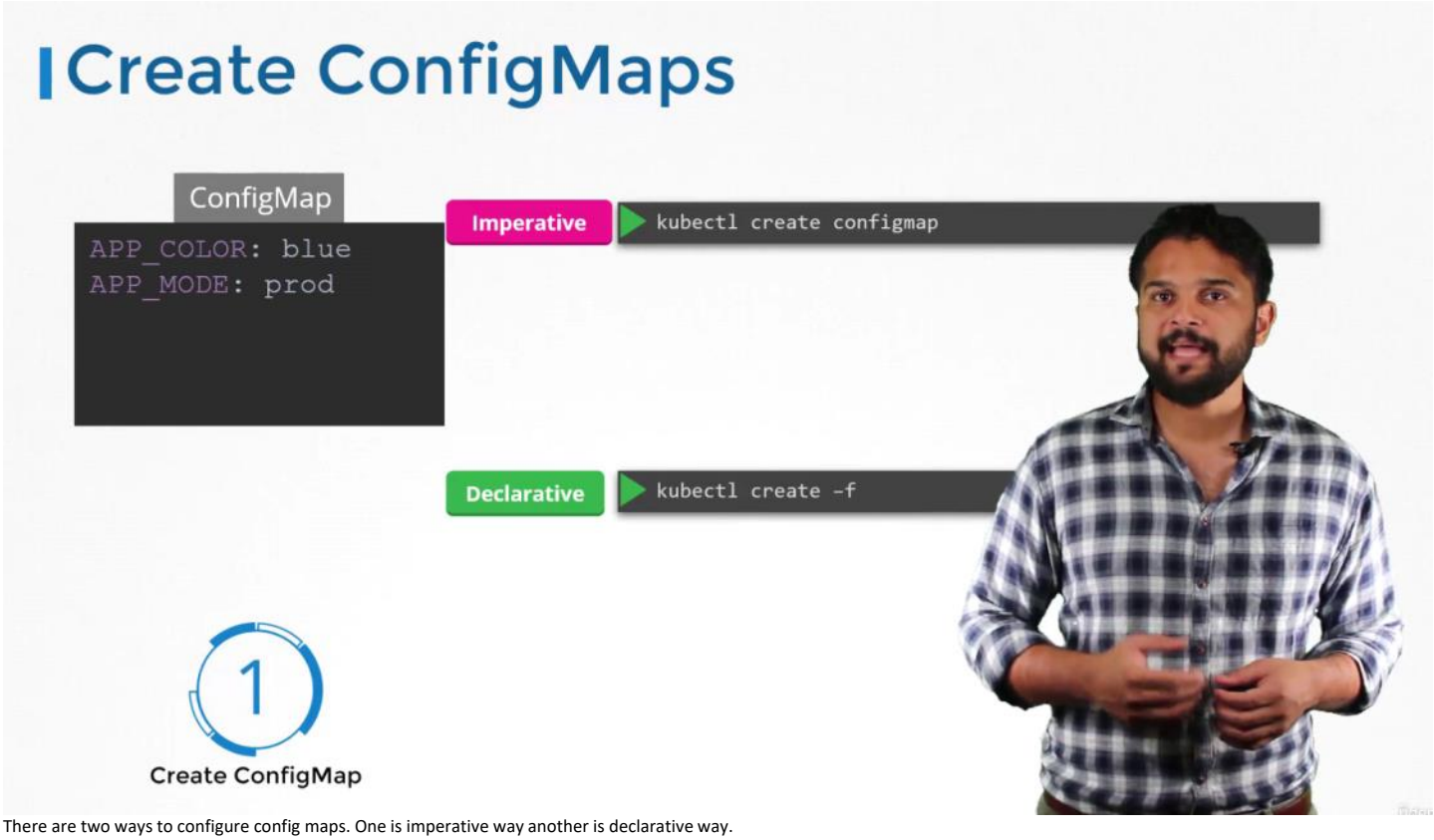
Wednesday, June 19, 2024 11:58 PM



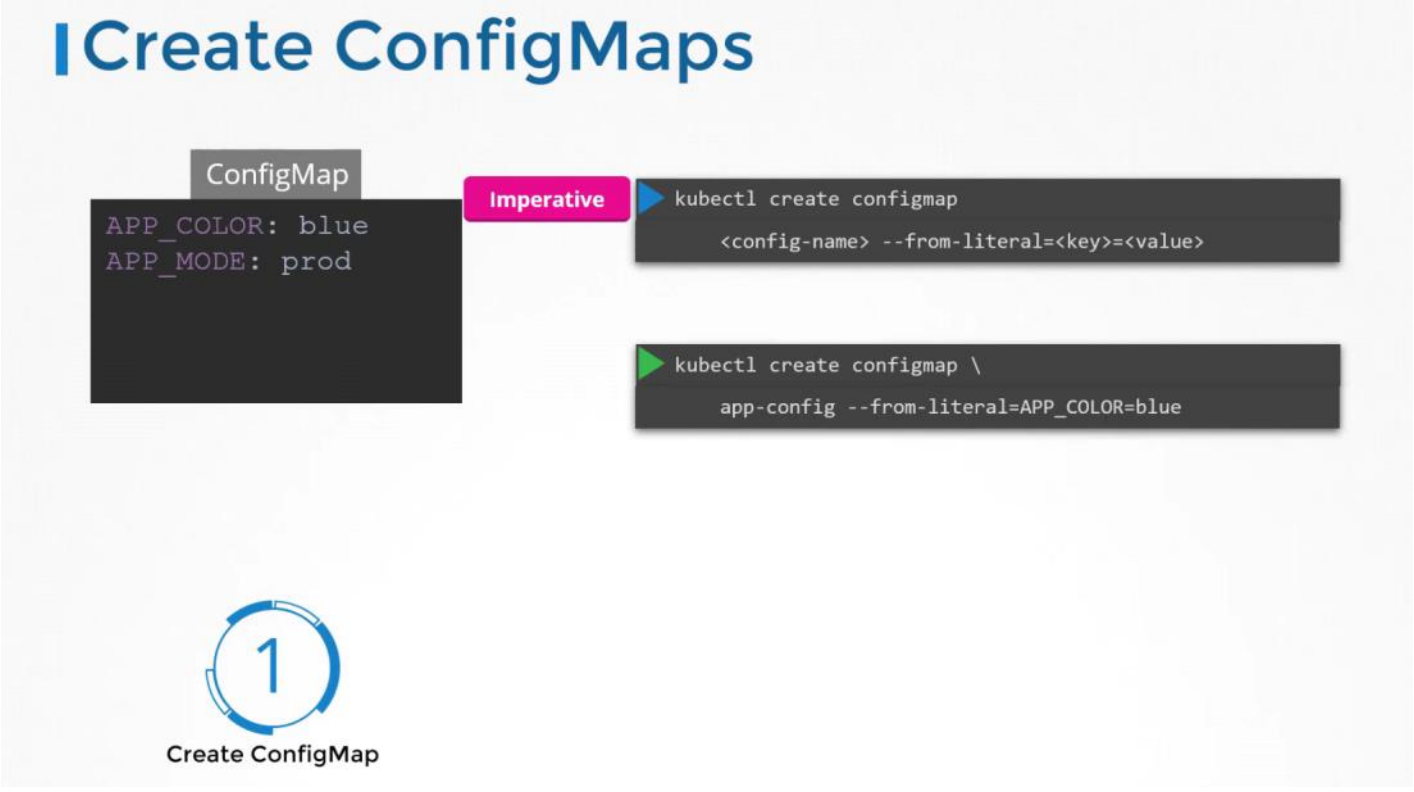
Three ways to set variables
config maps pass data as key value pair

Config maps in applications

Thursday, June 20, 2024 12:01 AM



There are two ways to configure config maps. One is imperative way another is declarative way.



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_MOD=prod
```

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

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

1

Create ConfigMap

Create ConfigMaps

ConfigMap

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

Declarative

```
kubectl create -f
```

config-map.yaml

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

1

Create ConfigMap

```
kubectl create -f config-map.yaml
```

Create ConfigMaps

app-config

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

mysql-config

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

redis-config

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

1

Create ConfigMap

View ConfigMaps

```
kubectl get configmaps
NAME         DATA      AGE
app-config   2         3s
```

```
kubectl describe configmaps
Name:         app-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
APP_COLOR:
----
blue
APP_MODE:
----
prod
Events:  <none>
```

ConfigMap in Pods

pod-definition.yaml

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

config-map.yaml

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

2

Inject into Pod

```
kubectl create -f pod-definition.yaml
```

ConfigMap in Pods

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

ENV

SINGLE ENV

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

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

VOLUME

you can inject configuration data as a single env var or as a volume

Practice- env variables

Thursday, June 20, 2024 11:35 PM

1 2 3 4 5 6 7 8 9 10 11

How many PODs exist on the system?
in the current(default) namespace

0

1

2

3

4

5

```
controlplane ~ ➜  k get pods
NAME           READY   STATUS    RESTARTS   AGE
webapp-color   1/1     Running   0          16s

controlplane ~ ➜
```

Task Hint 59:03 Terminal 1 +

1 2 3 4 5 6 7 8 9 10 11

What is the environment variable name set on the container in the pod?

COLOR

Sleep

APP_COLOR

APP-COLOR

APP

pink

```
webapp-color   1/1     Running   0          16s

controlplane ~ ➜  k describe pod webapp-color
Name:         webapp-color
Namespace:    default
Priority:     0
Node:         controlplane/172.25.0.34
Start Time:   Sat, 16 Apr 2022 22:49:43 +0000
Labels:       name=webapp-color
Annotations:  <none>
Status:       Running
IP:           10.42.0.9
IPs:
  IP:  10.42.0.9
Containers:
  webapp-color:
    Container ID:   containerd://47c9868ccadf518105084f41731db1eda50333911f308c1caf41d78595b28722
    Image:          kodekloud/webapp-color
    Image ID:       docker.io/kodekloud/webapp-color@sha256:99c3821ea49b89c7a22d3eebab5c2e1ec651452e7675
af243485034a72eb1423
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sat, 16 Apr 2022 22:49:49 +0000
    Ready:          True
    Restart Count:  0
    Environment:
      APP_COLOR:  pink
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-2nbfl (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-2nbfl:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
```

1 2 3 4 5 6 7 8 9 10 11

What is the value set on the environment variable `APP_COLOR` on the container in the pod?

COLOR

Sleep

green

APP-COLOR

blue

```
webapp-color   1/1     Running   0          16s

controlplane ~ ➜  k describe pod webapp-color
Name:         webapp-color
Namespace:    default
Priority:     0
Node:         controlplane/172.25.0.34
Start Time:   Sat, 16 Apr 2022 22:49:43 +0000
Labels:       name=webapp-color
Annotations:  <none>
Status:       Running
IP:           10.42.0.9
IPs:
  IP:  10.42.0.9
Containers:
  webapp-color:
    Container ID:   containerd://47c9868ccadf518105084f41731db1eda50333911f308c1caf41d78595b28722
    Image:          kodekloud/webapp-color
    Image ID:       docker.io/kodekloud/webapp-color@sha256:99c3821ea49b89c7a22d3eebab5c2e1ec651452e7675
af243485034a72eb1423
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sat, 16 Apr 2022 22:49:49 +0000
    Ready:          True
    Restart Count:  0
    Environment:
      APP_COLOR:  pink
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-2nbfl (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
```

1 2 3 4 5 6 7 8 9 10 11

Update the environment variable on the POD to display a `green` background

Note: Delete and recreate the POD. Only make the necessary changes. Do not modify the name of the Pod.

Check

- Pod Name: webapp-color
- Label Name: webapp-color
- Env: APP_COLOR=green

```
controlplane ~ ➜  kubectl edit pod webapp-color
error: pods "webapp-color" is invalid
A copy of your changes has been stored to "/tmp/kubectl-edit-3135302771.yaml"
error: Edit cancelled, no valid changes were saved.

controlplane ~ ✖ cat /tmp/kubectl-edit-3135302771.yaml
```

```
controlplane ~ ➜  kubectl replace --force -f /tmp/kubectl-edit-3135302771.yaml
pod "webapp-color" deleted
pod/webapp-color replaced

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

How many `ConfigMaps` exists in the `default` namespace?

0

1

3

4

5

```
controlplane ~ ➜  k get configmap
NAME                 DATA   AGE
kube-root-ca.crt     1      16m
db-config            3      22s

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

Identify the database host from the config map `db-config`

SQL01

SQL01.example.com

3306

MY-SQL-SERVER

DB-Host-01

```
NAME                 DATA   AGE
kube-root-ca.crt     1      16m
db-config            3      22s

controlplane ~ ➜  k describe cm db-config
Name:         db-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
DB_HOST:
----
SQL01.example.com
DB_NAME:
----
SQL01
DB_PORT:
----
3306

BinaryData
====

Events:  <none>

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

Create a new ConfigMap for the `webapp-color` POD. Use the spec given below.

Check

- ConfigName Name: webapp-config-map
- Data: APP_COLOR=darkblue

```
annotation. Otherwise, the annotation will be unchanged. This flag is useful when you want to
perform kubectl apply on this object in the future.
      --show-managed-fields=false: If true, keep the managedFields when printing objects in JSON or
YAML format.
      --template='': Template string or path to template file to use when -o=go-template,
-o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
      --validate=true: If true, use a schema to validate the input before sending it

Usage:
  kubectl create configmap NAME [--from-file=[key=]source] [--from-literal=key1=value1]
[--dry-run=server|client|none] [options]

Use "kubectl options" for a list of global command-line options (applies to all commands).

controlplane ~ ➜  kubectl create configmap webapp-config-map --from-literal=APP_COLOR=darkblue
configmap/webapp-config-map created

controlplane ~ ➜

controlplane ~ ➜

controlplane ~ ➜  k describe cm webapp-config-map
Name:         webapp-config-map
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
APP_COLOR:
----
darkblue

BinaryData
====

Events:  <none>
```

Use create cm --help for this question

kubectl create configmap --from-literal=APP_COLOR=darkblue

1 2 3 4 5 6 7 8 9 10 11

Update the environment variable on the POD to use the newly created ConfigMap
Note: Delete and recreate the POD. Only make the necessary changes. Do not modify the name of the Pod.

Check

- Pod Name: webapp-color
- EnvFrom: webapp-config-map

```
controlplane ~ ➜  kubectl edit pod webapp-color
error: pods "webapp-color" is invalid
A copy of your changes has been stored to "/tmp/kubectl-edit-1925296777.yaml"
error: Edit cancelled, no valid changes were saved.

controlplane ~ ✖
```

`

1 2 3 4 5 6 7 8 9 10 11

Update the environment variable on the POD to use the newly created ConfigMap
Note: Delete and recreate the POD. Only make the necessary changes. Do not modify the name of the Pod.

Check

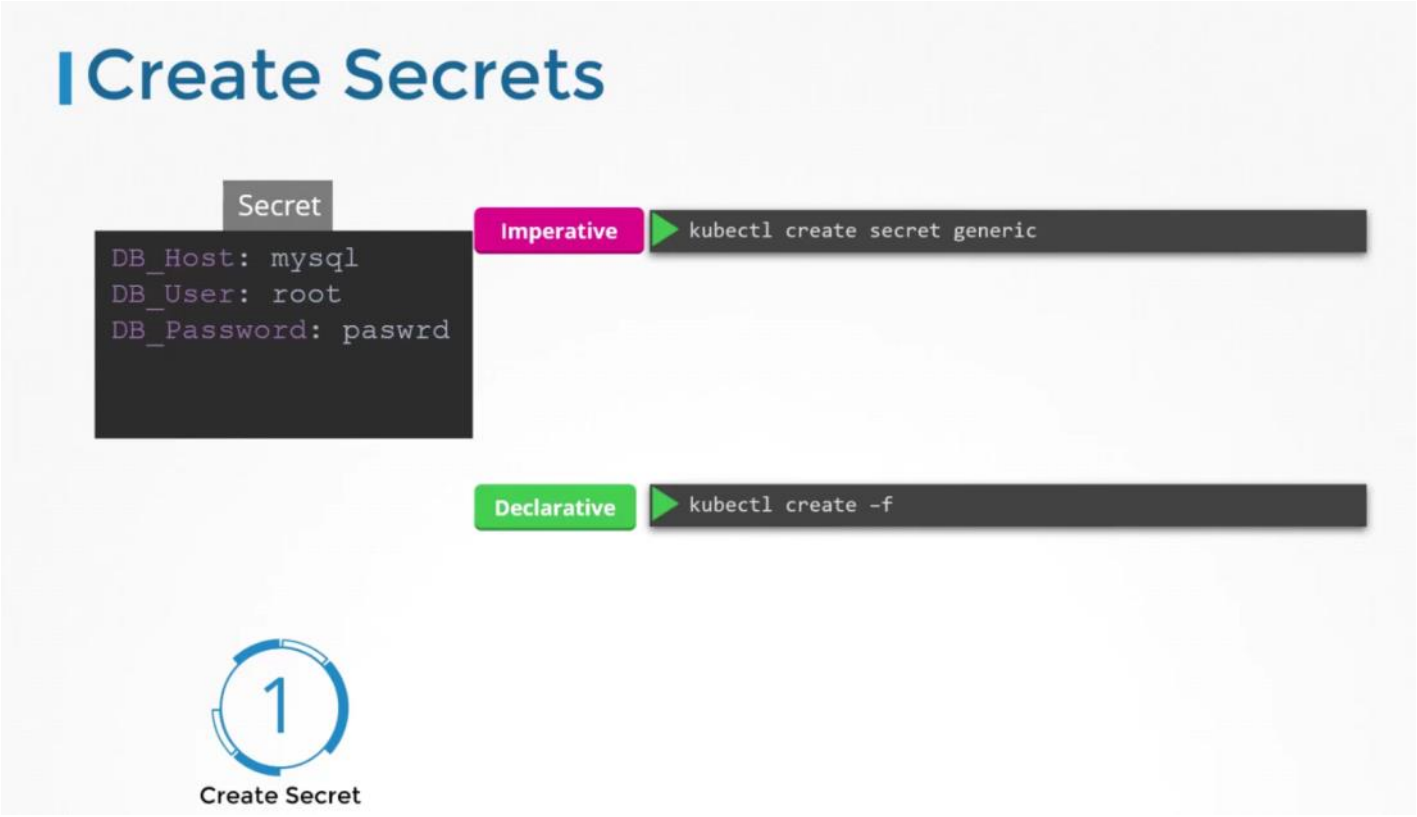
- Pod Name: webapp-color
- EnvFrom: webapp-config-map

```
# Please edit the object below. Lines beginning with a '#' will be
# and an empty file will abort the edit. If an error occurs while
# reopened with the relevant failures.
#
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2022-04-16T22:52:40Z"
  labels:
    name: webapp-color
  name: webapp-color
  namespace: default
  resourceVersion: "929"
  uid: f17244ec-f17f-4027-9f6c-58aa47f2cf03
spec:
  containers:
  - envFrom:
      - configMapRef:
          name: webapp-config-map
    image: kodekloud/webapp-color
    imagePullPolicy: Always
    name: webapp-color
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-2nbfl
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: controlplane
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
-- INSERT (paste) --
```

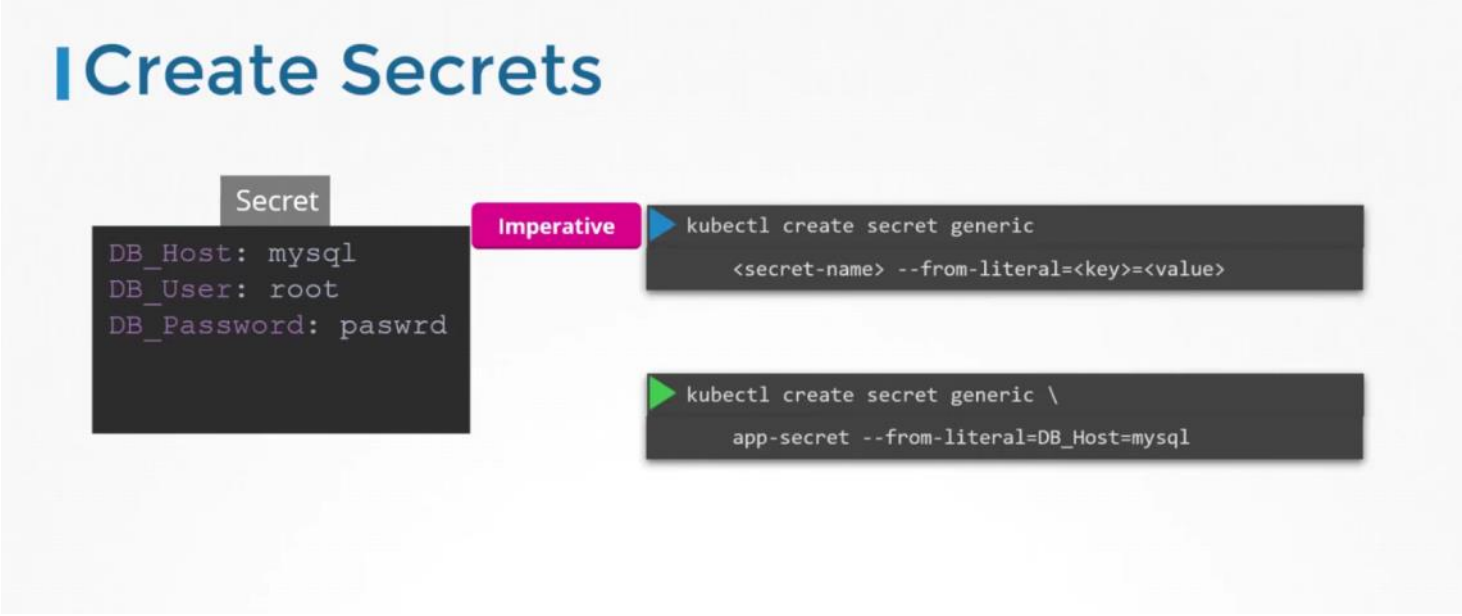
```
controlplane ~ ✖ kubectl replace --force -f /tmp/kubectl-edit-1925296777.yaml
pod "webapp-color" deleted
pod/webapp-color replaced

controlplane ~ ➜
```

Secrets in Kubernetes

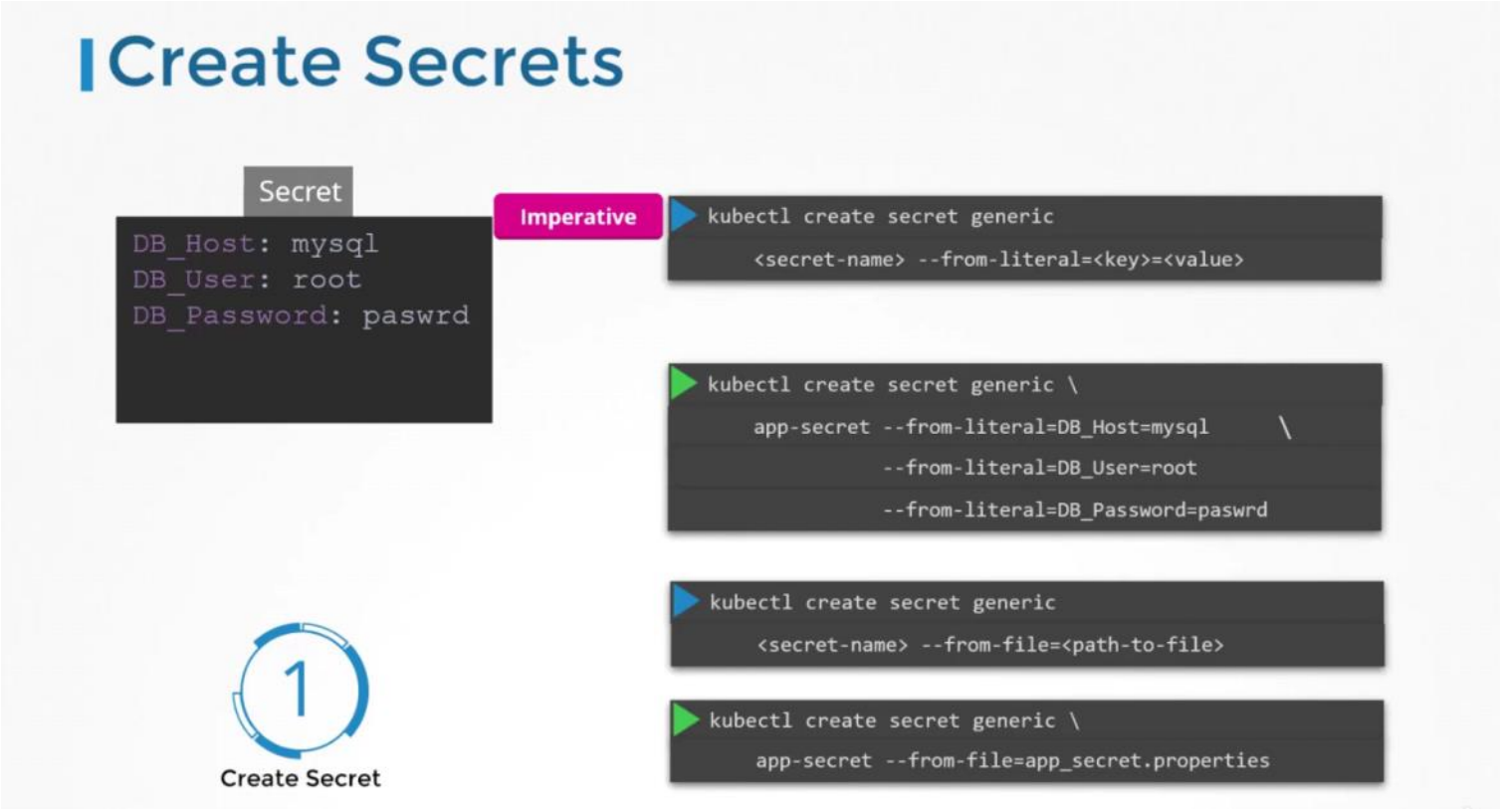
Sunday, June 23, 2024 8:16 PM



declarative way works by creating a secret definition file

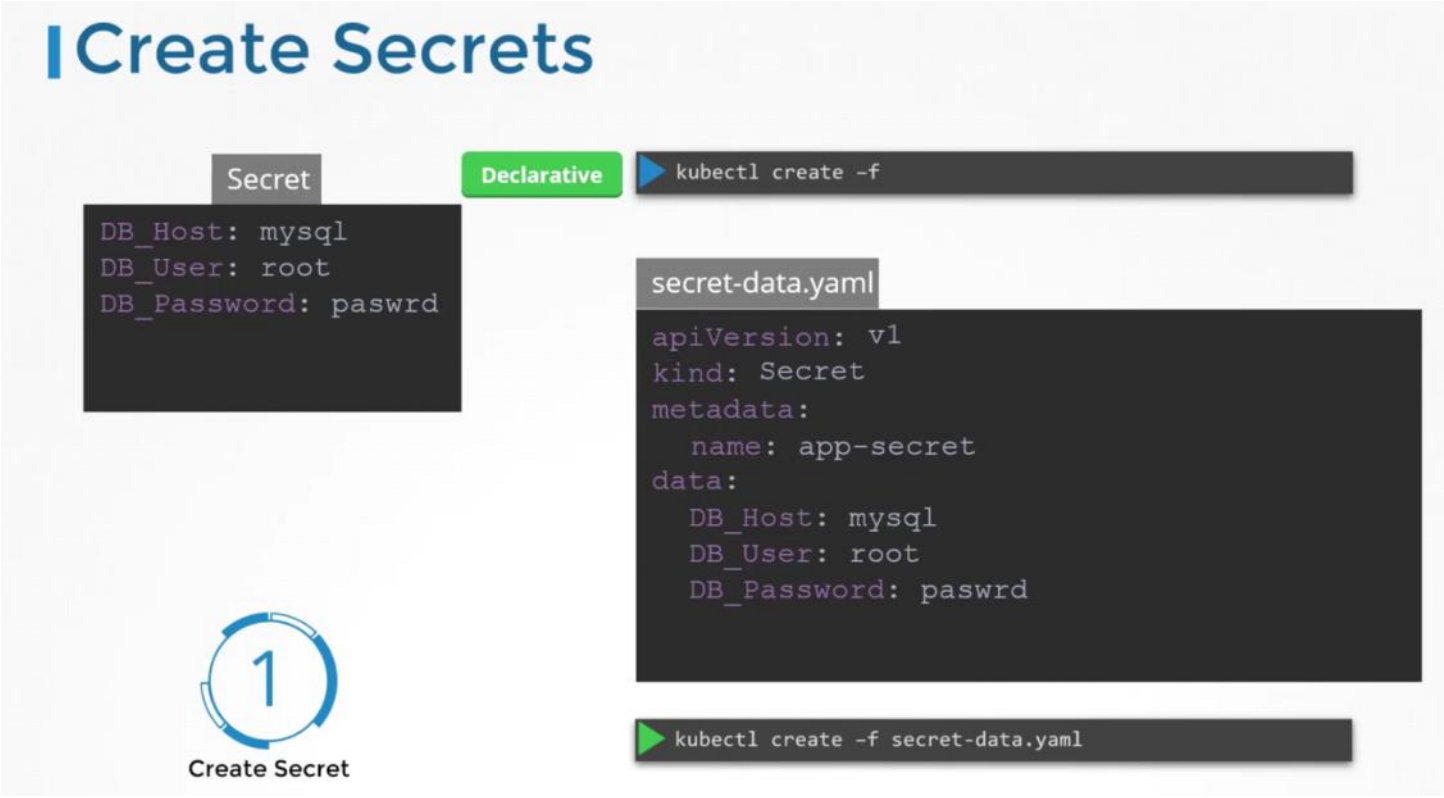


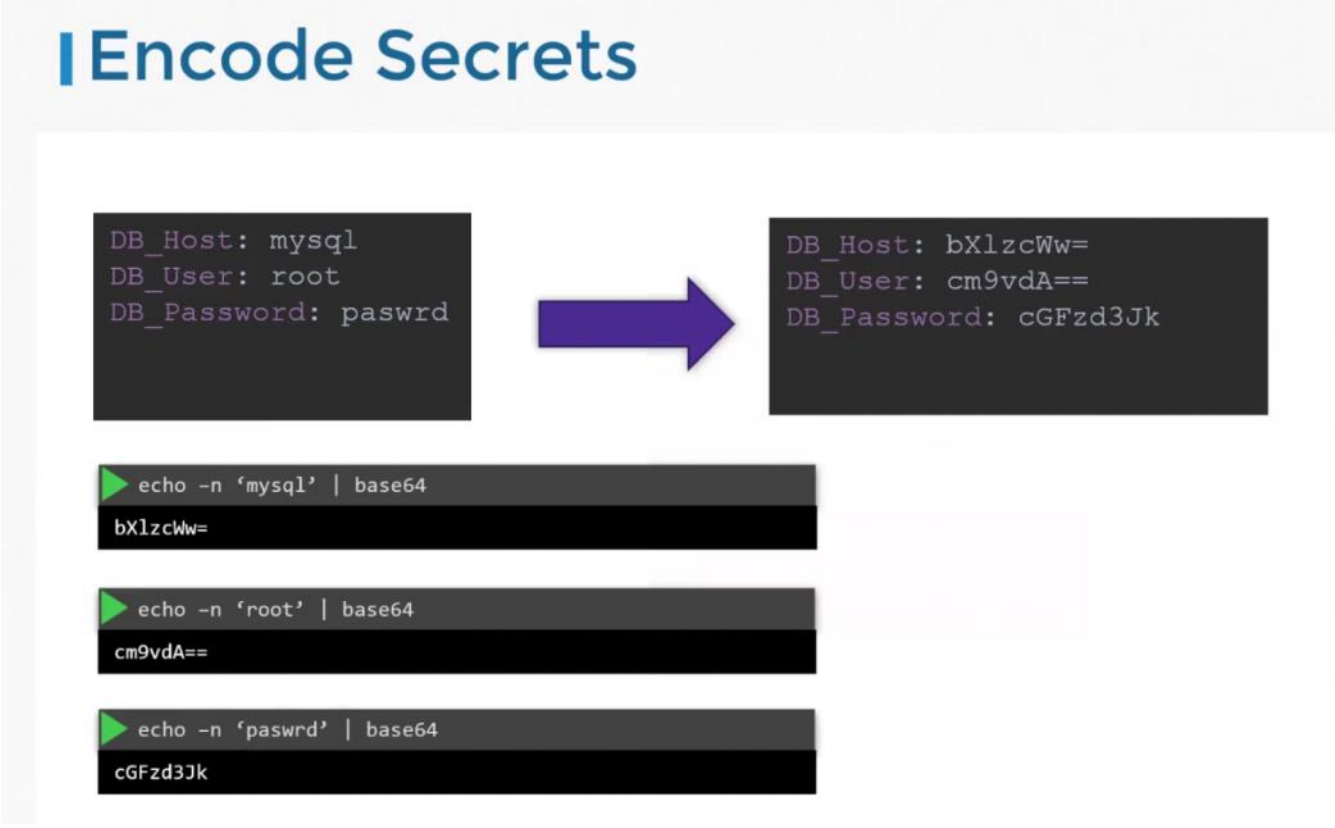
```
kubectl create secret generic \
    app-secret --from-literal=DB_Host=mysql      \
               --from-literal=DB_User=root
               --from-literal=DB_Password=paswrd
```



kubectl create secret generic db-secret --from-literal=DB_Hos
t=sql01 --from-literal=DB_User=root --from-literal=DB_Password=password123
secret/db-secret created

Declarative way





to encode the secrets run the echo -n command on linux

View Secrets

```
kubectl get secrets
NAME                  TYPE                                  DATA      AGE
app-secret            Opaque                                3         10m
default-token-mvtkv   kubernetes.io/service-account-token   3         2h
```

```
kubectl describe secrets
Name:         app-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
DB_Host:      10 bytes
DB_Password:  6 bytes
DB_User:      4 bytes
```

View Secrets

```
kubectl get secrets
NAME                  TYPE                                  DATA      AGE
app-secret            Opaque                                3         10m
default-token-mvtkv   kubernetes.io/service-account-token   3         2h
```

```
kubectl describe secrets
Name:         app-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
DB_Host:      10 bytes
DB_Password:  6 bytes
DB_User:      4 bytes
```

```
kubectl get secret app-secret -o yaml
apiVersion: v1
data:
  DB_Host: bXlzcWw=
  DB_Password: cGFzd3Jk
  DB_User: cm9vdA==
kind: Secret
metadata:
  creationTimestamp: 2018-10-18T10:01:12Z
  labels:
    name: app-secret
  name: app-secret
  namespace: default
uid: be96e989-d2bc-11e8-a545-080027931072
type: Opaque
```

to view the attributes values

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
```

to decode the secrets you will have to run the same command but with the decode option

Secrets in Pods

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:
      - secretRef:
          name: app-secret
```

secret-data.yaml

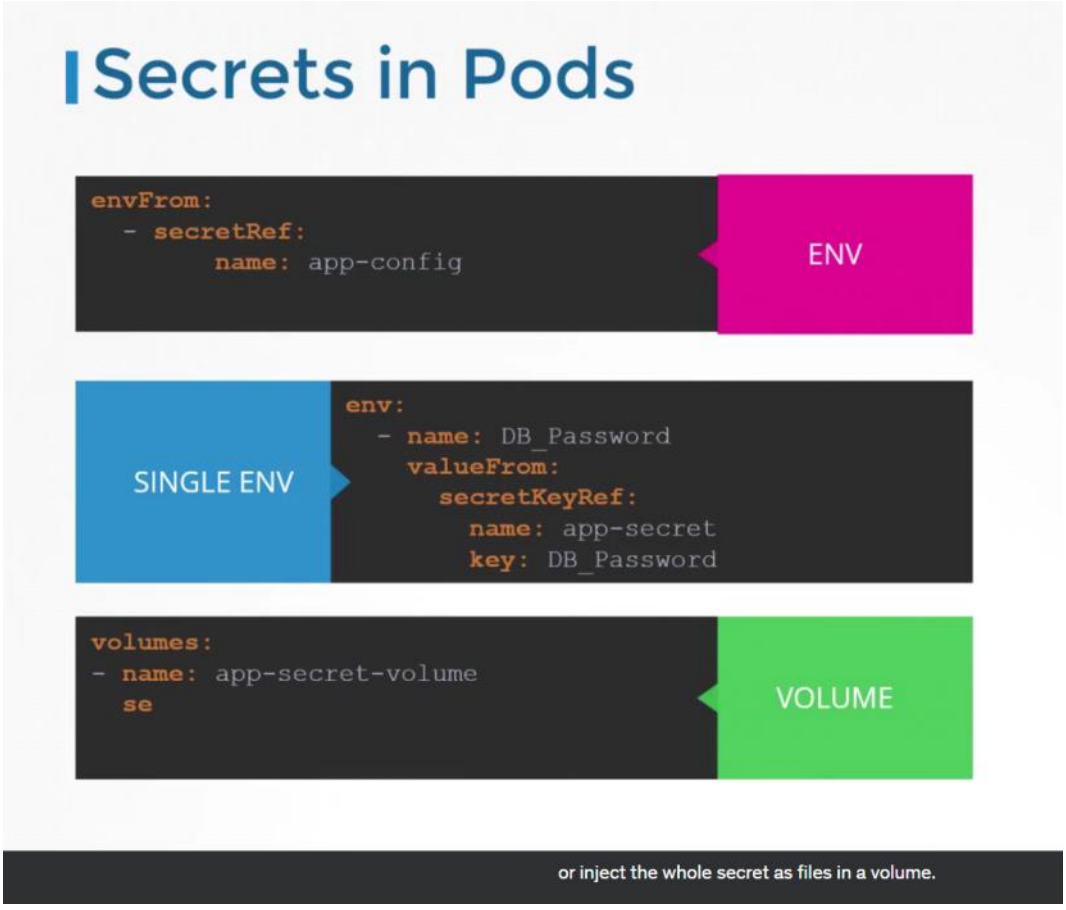
```
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
data:
  DB_Host: bXlzcWw=
  DB_User: cm9vdA==
  DB_Password: cGFzd3Jk
```

2

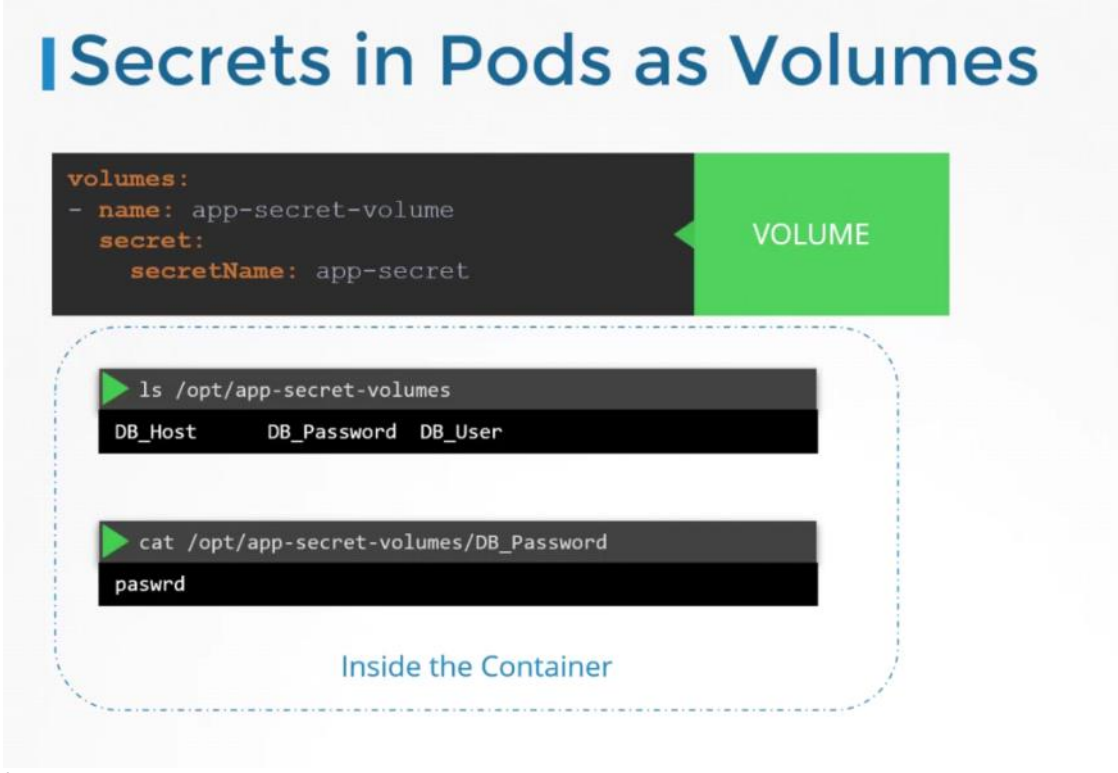
Inject into Pod

```
kubectl create -f pod-definition.yaml
```

to configure it with a pod write env from property and specify the name of the secret we created earlier
this is called injecting secrets as env variable.



Other ways to inject secrets in pods are by using single env variable or injecting secret as a file in a volume.



f

if injecting secret as a volume you will have to create separate file for each secret and contains value of the secret as its content

Practice- secrets

Monday, June 24, 2024 9:30 PM

1 2 3 4 5 6 7 8

How many `Secrets` exist on the system?
in the current(default) namespace

0

1

2

3

4

```
controlplane ~ ➜  kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-cr4sr   kubernetes.io/service-account-token   3      7m50s

controlplane ~ ➜
```

1 2 3 4 5 6 7 8

How many secrets are defined in the `default-token` secret?

0

1

2

3

4

5

```
controlplane ~ ➜  kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-cr4sr   kubernetes.io/service-account-token   3      7m50s

controlplane ~ ➜  k describe secret default-token-cr4sr
Name:         default-token-cr4sr
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: default
              kubernetes.io/service-account.uid: 2751dc84-936d-4b41-86d5-8427a99a2b2b

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     570 bytes
namespace:  7 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6ImpEYmFtckZBTXdsZ21LLTRjLXJVaWpzWEpzMU9ta2xRUER5VEJ6d0w5SEUifQ.e
yJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZ
WZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImRlZmF1bHQtdG9rZW4tY3I0c3IiLCJrdWJlc
m5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiZGVmYXVsdCIsImt1YmVybmV0ZXMuaW8vc2VydmljZ
WFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6IjI3NTFkYzg0LTkzNmQtNGI0MS04NmQ1LTg0MjdhOTlhMmIyYiIsInN1YiI6InN5c
3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.FzvS-7_aJWRBG-3UUh12deJE8ow-f5tdEqRNkh4L_3fdCvI7sokAnN
DupWo4g8gGlNesoqENLwbgrw_e6NdUxHSwUVyPvwlbSx1p2jNhhY6UxSFGNbhqMsvJar_jMX9DYjlZNH5WAfN4e9CsFlh5eJoUavBsa7
Mc6fgkMSpTa25m47hD1poEPatYR4YuVyweJy3aDQR-eHfjq_xjVzAFlifvvYB4jpNI1FgyYkmQbUtwg67uP0cQWaKivfF-k0pIIHAhPs
UXqRq8I5-svoj7nYxBbxCOP5Zrp1Nz6R4wOxi6vYC_VJh_pp1i8b688k3aB2Y4EB28zYz4R358jqrBLw

controlplane ~ ➜
```

1 2 3 4 5 6 7 8

What is the type of the `default-token` secret?

- Secret
- Opaque
- kubernetes.io/service-account-token

```
controlplane ~ ➜  kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-cr4sr   kubernetes.io/service-account-token   3      7m50s

controlplane ~ ➜  k describe secret default-token-cr4sr
Name:         default-token-cr4sr
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: default
              kubernetes.io/service-account.uid: 2751dc84-936d-4b41-86d5-8427a99a2b2b

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     570 bytes
namespace:  7 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6ImpEYmFtckZBTXdsZ21LLTRjLXJVaWpzWEpzMU9ta2xRUER5VEJ6d0w5SEUifQ.e
yJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZ
WZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImRlZmF1bHQtdG9rZW4tY3I0c3IiLCJrdWJlc
m5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiZGVmYXVsdCIsImt1YmVybmV0ZXMuaW8vc2VydmljZ
WFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6IjI3NTFkYzg0LTkzNmQtNGI0MS04NmQ1LTg0MjdhOTlhMmIyYiIsInN1YiI6InN5c
3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.FzvS-7_aJWRBG-3UUh12deJE8ow-f5tdEqRNkh4L_3fdCvI7sokAnN
DupWo4g8gGlNesoqENLwbgrw_e6NdUxHSwUVyPvwlbSx1p2jNhhY6UxSFGNbhqMsvJar_jMX9DYjlZNH5WAfN4e9CsFlh5eJoUavBsa7
Mc6fgkMSpTa25m47hD1poEPatYR4YuVyweJy3aDQR-eHfjq_xjVzAFlifvvYB4jpNI1FgyYkmQbUtwg67uP0cQWaKivfF-k0pIIHAhPs
UXqRq8I5-svoj7nYxBbxCOP5Zrp1Nz6R4wOxi6vYC_VJh_pp1i8b688k3aB2Y4EB28zYz4R358jqrBLw

controlplane ~ ➜
```

1 2 3 4 5 6 7 8

Which of the following is not a secret data defined in `default-token` secret?

- namespace
- ca.crt
- token
- type

```
controlplane ~ ➜  kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-cr4sr   kubernetes.io/service-account-token   3      7m50s

controlplane ~ ➜  k describe secret default-token-cr4sr
Name:         default-token-cr4sr
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: default
              kubernetes.io/service-account.uid: 2751dc84-936d-4b41-86d5-8427a99a2b2b

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     570 bytes
namespace:  7 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6ImpEYmFtckZBTXdsZ21LLTRjLXJVaWpzWEpzMU9ta2xRUER5VEJ6d0w5SEUifQ.e
yJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZ
WZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImRlZmF1bHQtdG9rZW4tY3I0c3IiLCJrdWJlc
m5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiZGVmYXVsdCIsImt1YmVybmV0ZXMuaW8vc2VydmljZ
WFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6IjI3NTFkYzg0LTkzNmQtNGI0MS04NmQ1LTg0MjdhOTlhMmIyYiIsInN1YiI6InN5c
3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.FzvS-7_aJWRBG-3UUh12deJE8ow-f5tdEqRNkh4L_3fdCvI7sokAnN
DupWo4g8gGlNesoqENLwbgrw_e6NdUxHSwUVyPvwlbSx1p2jNhhY6UxSFGNbhqMsvJar_jMX9DYjlZNH5WAfN4e9CsFlh5eJoUavBsa7
Mc6fgkMSpTa25m47hD1poEPatYR4YuVyweJy3aDQR-eHfjq_xjVzAFlifvvYB4jpNI1FgyYkmQbUtwg67uP0cQWaKivfF-k0pIIHAhPs
UXqRq8I5-svoj7nYxBbxCOP5Zrp1Nz6R4wOxi6vYC_VJh_pp1i8b688k3aB2Y4EB28zYz4R358jqrBLw

controlplane ~ ➜
```

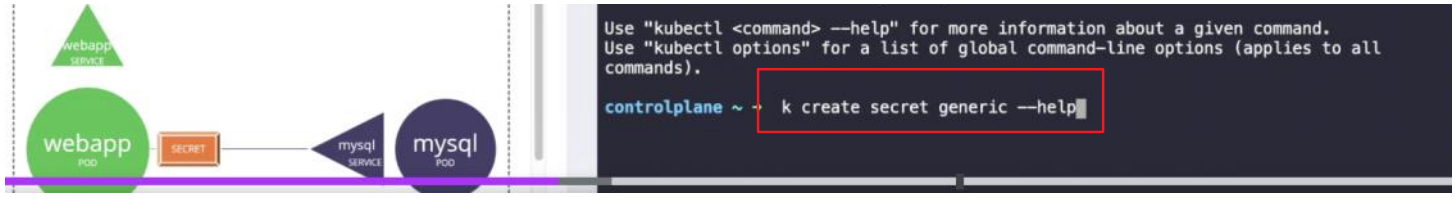
1 2 3 4 5 6 7 8

The reason the application is failed is because we have not created the secrets yet. Create a new secret named `db-secret` with the data given below.
You may follow any one of the methods discussed in lecture to create the secret.

Check

- Secret Name: db-secret
- Secret 1: DB_Host=sql01
- Secret 2: DB_User=root
- Secret 3: DB_Password=password123

```
controlplane ~ ➜  k get deploy
No resources found in default namespace.

controlplane ~ ➜  k get pods
NAME         READY   STATUS    RESTARTS   AGE
webapp-pod   1/1     Running   0          26s
mysql        1/1     Running   0          26s

controlplane ~ ➜  k get svc
NAME             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
kubernetes       ClusterIP   10.43.0.1      <none>        443/TCP          10m
webapp-service   NodePort    10.43.73.93    <none>        8080:30080/TCP   40s
sql01            ClusterIP   10.43.128.20   <none>        3306/TCP         40s

controlplane ~ ➜  k get secret
NAME                  TYPE                                  DATA   AGE
default-token-cr4sr   kubernetes.io/service-account-token   3      10m

controlplane ~ ➜  k create secret --help
Create a secret using specified subcommand.

Available Commands:
  docker-registry Create a secret for use with a Docker registry
  generic         Create a secret from a local file, directory, or literal value
  tls             Create a TLS secret

Usage:
  kubectl create secret [flags] [options]

Use "kubectl <command> --help" for more information about a given command.
Use "kubectl options" for a list of global command-line options (applies to all
commands).
```

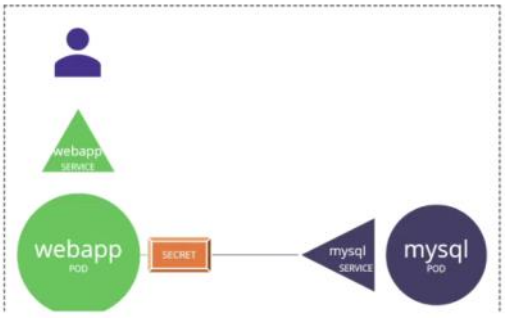


1 2 3 4 5 6 7 8

The reason the application is failed is because we have not created the secrets yet. Create a new secret named `db-secret` with the data given below.
You may follow any one of the methods discussed in lecture to create the secret.

Complete Next

- Secret Name: db-secret
- Secret 1: DB_Host=sql01
- Secret 2: DB_User=root
- Secret 3: DB_Password=password123




```
controlplane ~ ➜  k create secret generic db-secret --from-literal=DB_Host=sql01 --from-li
teral DB_User=root --from-literal DB_Password=password123
secret/db-secret created

controlplane ~ ➜  k get secret
NAME                  TYPE                                  DATA   AGE
default-token-cr4sr   kubernetes.io/service-account-token   3      12m
db-secret             Opaque                                3      6s

controlplane ~ ➜  k describe secret db-secret
Name:         db-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
DB_Password:  11 bytes
DB_User:      4 bytes
DB_Host:      5 bytes

controlplane ~ ➜
```

1 2 3 4 5 6 7 8

Configure `webapp-pod` to load environment variables from the newly created secret.
Delete and recreate the pod if required.

Check

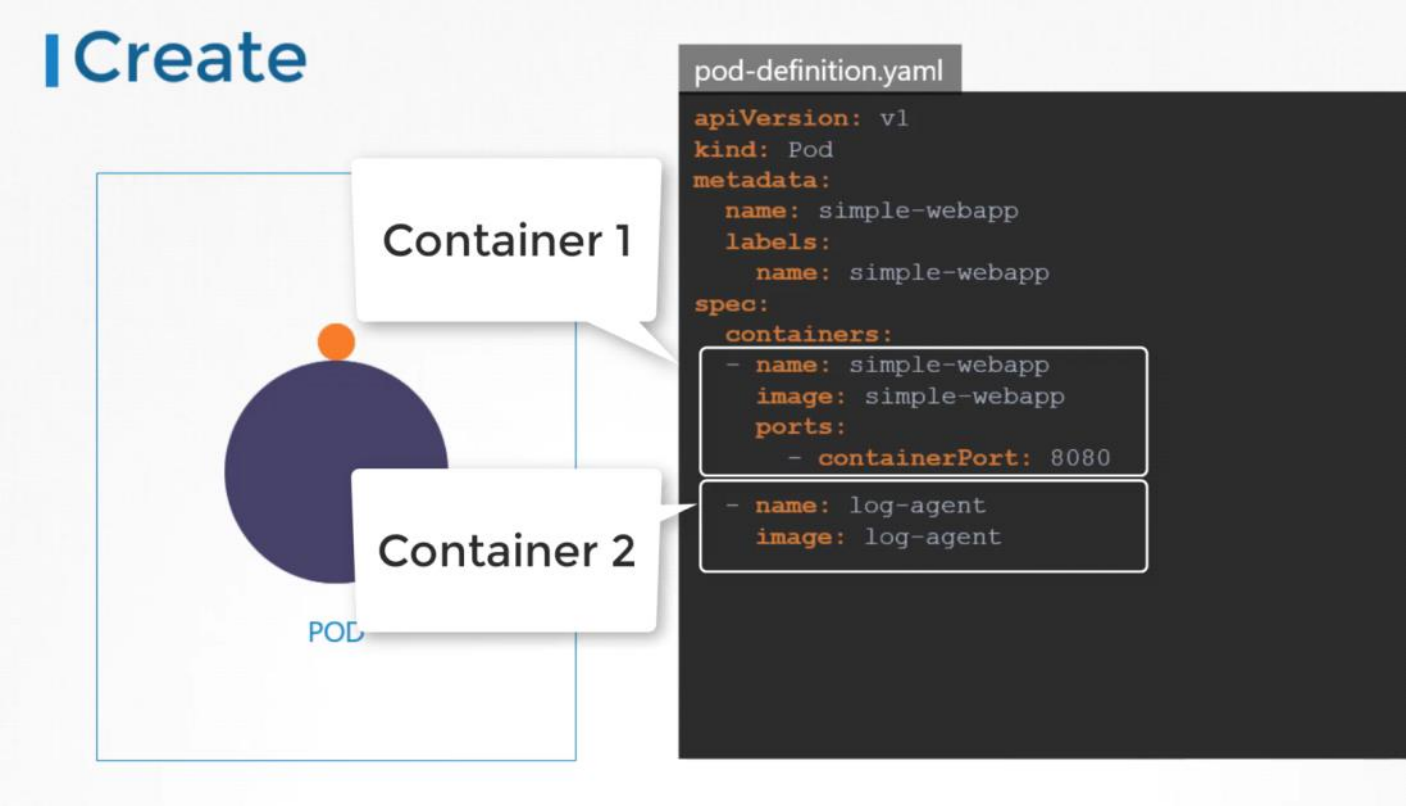
- Pod name: webapp-pod
- Image name: kodekloud/simple-webapp-mysql
- Env From: Secret=db-secret

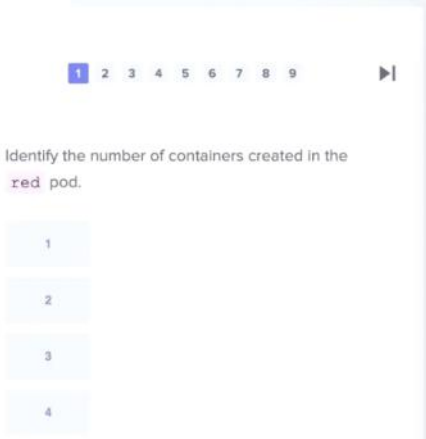
```
# and an empty file will abort the edit. If an error occurs while saving this file will b
# reopened with the relevant failures.
#
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2022-04-17T17:28:58Z"
  labels:
    name: webapp-pod
  name: webapp-pod
  namespace: default
  resourceVersion: "822"
  uid: 98e92f71-36e8-480d-b114-b1dc3848291c
spec:
  containers:
  - image: kodekloud/simple-webapp-mysql
    imagePullPolicy: Always
    name: webapp
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    envFrom:
      - secretRef:
          name: db-secret
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-dxllf
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: controlplane
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
```

```
controlplane ~ ➜  kubectl edit pod webapp-pod
error: pods "webapp-pod" is invalid
A copy of your changes has been stored to "/tmp/kubectl-edit-123695794.yaml"
error: Edit cancelled, no valid changes were saved.

controlplane ~ ✖ kubectl replace --force -f /tmp/kubectl-edit-123695794.yaml
pod "webapp-pod" deleted
pod/webapp-pod replaced

controlplane ~ ➜

controlplane ~ ➜

controlplane ~ ➜  cle
```

Multi Container Pods

Monday, June 24, 2024 11:49 PM






```
Containers:
  apple:
    Container ID:
    Image:          busybox
    Image ID:
    Port:           <none>
    Host Port:      <none>
    Command:
      sleep
      4500
    State:           Waiting
      Reason:        ContainerCreating
    Ready:           False
    Restart Count:   0
    Environment:     <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-7sggv (ro)
  wine:
    Container ID:
    Image:          busybox
    Image ID:
    Port:           <none>
    Host Port:      <none>
    Command:
      sleep
      4500
    State:           Waiting
      Reason:        ContainerCreating
    Ready:           False
    Restart Count:   0
    Environment:     <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-7sggv (ro)
  scarlet:
    Container ID:
    Image:          busybox
    Image ID:
    Port:           <none>
    Host Port:      <none>
    Command:
```

1 2 3 4 5 6 7 8 9

Identify the name of the containers running in the `blue` pod.

Red & green

orange & yellow

grey & white

```
Node:         controlplane/10.40.119.6
Start Time:   Sun, 17 Apr 2022 18:16:47 +0000
Labels:       <none>
Annotations:  <none>
Status:       Pending
IP:
IPs:          <none>
Containers:
  teal:
    Container ID:
    Image:          busybox
    Image ID:
    Port:           <none>
    Host Port:      <none>
    Command:
      sleep
      4500
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-6qm72 (ro)
  navy:
    Container ID:
    Image:          busybox
    Image ID:
    Port:           <none>
    Host Port:      <none>
    Command:
      sleep
      4500
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-6qm72 (ro)
```

1 2 3 4 5 6 7 8 9

Create a multi-container pod with 2 containers.
Use the spec given below.

If the pod goes into the `crashloopbackoff`

then add `sleep 1000` in the `lemon` container.

Check

- Name: yellow
- Container 1 Name: lemon
- Container 1 Image: busybox
- Container 2 Name: gold
- Container 2 Image: redis

```
root@controlplane ~ ➜  k run yellow --image=busybox --dry-run=client -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: yellow
  name: yellow
spec:
  containers:
  - image: busybox
    name: yellow
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}

root@controlplane ~ ➜  k run yellow --image=busybox --dry-run=client -o yaml > yellow.yaml

root@controlplane ~ ➜

root@controlplane ~ ➜  vi yellow.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: yellow
  name: yellow
spec:
  containers:
  - image: busybox
    name: lemon
    resources: {}
  - name: gold
    image: redis
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
~
~
~
~
```

1 2 3 4 5 6 7 8 9

Create a multi-container pod with 2 containers.
Use the spec given below.

If the pod goes into the `crashloopbackoff`

then add `sleep 1000` in the `lemon` container.

Check

- Name: yellow
- Container 1 Name: lemon
- Container 1 Image: busybox
- Container 2 Name: gold
- Container 2 Image: redis

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: yellow
  name: yellow
spec:
  containers:
  - image: busybox
    name: lemon
    resources: {}
    command: [ "sleep", "1000" ]
  - name: gold
    image: redis
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

1 2 3 4 5 6 7 8 9

Create a multi-container pod with 2 containers.
Use the spec given below.

If the pod goes into the `crashloopbackoff`

then add `sleep 1000` in the `lemon` container.

Check

- Name: yellow
- Container 1 Name: lemon
- Container 1 Image: busybox
- Container 2 Name: gold
- Container 2 Image: redis

```
Node:         controlplane/10.40.119.6
Start Time:   Sun, 17 Apr 2022 18:19:30 +0000
Labels:       run=yellow
Annotations:  <none>
Status:       Pending
IP:
IPs:          <none>
Containers:
  lemon:
    Container ID:
    Image:          busybox
    Image ID:
    Port:           <none>
    Host Port:      <none>
    Command:
      sleep
      1000
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-pzm55 (ro)
  gold:
    Container ID:
    Image:          redis
    Image ID:
    Port:           <none>
    Host Port:      <none>
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-pzm55 (ro)
Conditions:
  Type              Status
  Initialized       True
```

1 2 3 4 5 6 7 8 9

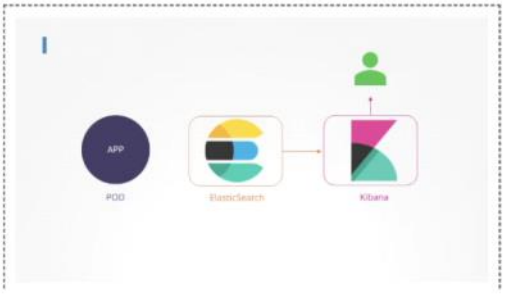
The application outputs logs to the file `/log/app.log`. View the logs and try to identify the user having issues with Login.
Inspect the log file inside the pod.

USER1

USER3

USER7

USER5




```
root@controlplane ~ ➜  k logs app -n elastic-stack
```

1 2 3 4 5 6 7 8 9

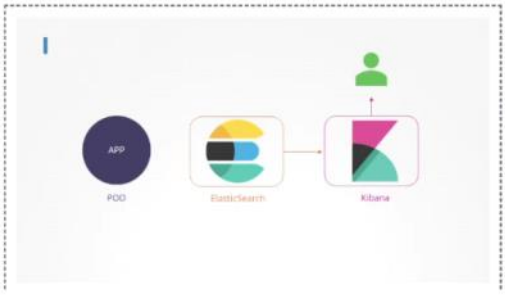
The application outputs logs to the file `/log/app.log`. View the logs and try to identify the user having issues with Login.
Inspect the log file inside the pod.

USER1

USER3

USER7

USER5




```
 is locked due to MANY FAILED ATTEMPTS.
[2022-04-17 18:22:38,771] INFO in event-simulator: USER1 logged out
[2022-04-17 18:22:39,774] INFO in event-simulator: USER4 logged in
[2022-04-17 18:22:40,775] INFO in event-simulator: USER2 is viewing page3
[2022-04-17 18:22:41,776] INFO in event-simulator: USER4 logged out
[2022-04-17 18:22:42,777] INFO in event-simulator: USER3 is viewing page2
[2022-04-17 18:22:43,779] WARNING in event-simulator: USER5 Failed to Login as the account
 is locked due to MANY FAILED ATTEMPTS.
[2022-04-17 18:22:43,779] INFO in event-simulator: USER4 is viewing page1
[2022-04-17 18:22:44,780] WARNING in event-simulator: USER7 Order failed as the item is OU
T OF STOCK.
[2022-04-17 18:22:44,781] INFO in event-simulator: USER1 is viewing page3
[2022-04-17 18:22:45,782] INFO in event-simulator: USER4 is viewing page1
[2022-04-17 18:22:46,783] INFO in event-simulator: USER4 logged out
[2022-04-17 18:22:47,784] INFO in event-simulator: USER4 is viewing page1
[2022-04-17 18:22:48,785] WARNING in event-simulator: USER5 Failed to Login as the account
 is locked due to MANY FAILED ATTEMPTS.
[2022-04-17 18:22:48,785] INFO in event-simulator: USER4 is viewing page3
[2022-04-17 18:22:49,785] INFO in event-simulator: USER1 is viewing page3
[2022-04-17 18:22:50,786] INFO in event-simulator: USER4 is viewing page3
[2022-04-17 18:22:51,787] INFO in event-simulator: USER4 is viewing page1
[2022-04-17 18:22:52,789] WARNING in event-simulator: USER7 Order failed as the item is OU
T OF STOCK.
[2022-04-17 18:22:52,789] INFO in event-simulator: USER4 is viewing page3
[2022-04-17 18:22:53,791] WARNING in event-simulator: USER5 Failed to Login as the account
 is locked due to MANY FAILED ATTEMPTS.
[2022-04-17 18:22:53,791] INFO in event-simulator: USER4 is viewing page3
[2022-04-17 18:22:54,792] INFO in event-simulator: USER3 logged in
[2022-04-17 18:22:55,794] INFO in event-simulator: USER3 logged in
[2022-04-17 18:22:56,795] INFO in event-simulator: USER4 logged out
[2022-04-17 18:22:57,796] INFO in event-simulator: USER2 logged out
[2022-04-17 18:22:58,798] WARNING in event-simulator: USER5 Failed to Login as the account
 is locked due to MANY FAILED ATTEMPTS.
[2022-04-17 18:22:58,798] INFO in event-simulator: USER1 is viewing page2
[2022-04-17 18:22:59,799] INFO in event-simulator: USER4 logged out
[2022-04-17 18:23:00,800] WARNING in event-simulator: USER7 Order failed as the item is OU
T OF STOCK.
[2022-04-17 18:23:00,801] INFO in event-simulator: USER2 logged in

root@controlplane ~ ➜
```

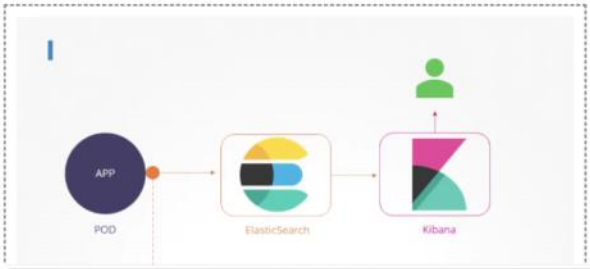
k -n elastic-stack exec -it app -- cat /log/app.log

1 2 3 4 5 6 7 8 9

Edit the pod to add a sidecar container to send logs to Elastic Search. Mount the log volume to the sidecar container.
Only add a new container. Do not modify anything else. Use the spec provided below.

Check

- Name: app
- Container Name: sidecar
- Container Image: kodekloud/filebeat-configured
- Volume Mount: log-volume
- Mount Path: /var/log/event-simulator/
- Existing Container Name: app
- Existing Container Image: kodekloud/event-simulator




```
root@controlplane ~ ➜  k edit pod app -n elastic-stack
error: pods "app" is invalid
A copy of your changes has been stored to "/tmp/kubectl-edit-3922970489.yaml"
error: Edit cancelled, no valid changes were saved.

root@controlplane ~ ✖ k replace --force -f /tmp/kubectl-edit-3922970489.yaml
pod "app" deleted
pod/app replaced

root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9

Identify the pod that has an `initContainer` configured.

red

green

blue

pink

```
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  49s   default-scheduler  Successfully assigned default/green to controlplane
  Normal  Pulling    48s   kubelet            Pulling image "busybox:1.28"
  Normal  Pulled     47s   kubelet            Successfully pulled image "busybox:1.28" in 1.291652583s
  Normal  Created    47s   kubelet            Created container green-container-1
  Normal  Started    46s   kubelet            Started container green-container-1
  Normal  Pulled     46s   kubelet            Container image "busybox:1.28" already present on machine
  Normal  Created    46s   kubelet            Created container green-container-2
  Normal  Started    46s   kubelet            Started container green-container-2


Name:         blue
Namespace:    default
Priority:     0
Node:         controlplane/172.25.0.75
Start Time:   Sun, 17 Apr 2022 18:51:51 +0000
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.42.0.11
IPs:
  IP:  10.42.0.11
Init Containers:
  init-myservice:
    Container ID:  containerd://28fa7c4e9d67b048794557903be8a61357eba3fa1e30568dcb3e30aa52adfbcd
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:d2b53584f580310186df7a2055ce3ff83cc0df6caacf1e3489bf
f8cf5d0af5d8
    Port:          <none>
    Host Port:     <none>
    Command:
```

1 2 3 4 5 6 7 8 9

What is the state of the `initContainer` on pod `blue`

Running

Terminated

Alive

Completed

```
controplane ~ ➜  k describe pod blue
Name:         blue
Namespace:    default
Priority:     0
Node:         controlplane/172.25.0.75
Start Time:   Sun, 17 Apr 2022 18:51:51 +0000
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.42.0.11
IPs:
  IP:  10.42.0.11
Init Containers:
  init-myservice:
    Container ID:  containerd://28fa7c4e9d67b048794557903be8a61357eba3fa1e30568dcb3e30aa52adfbcd
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:d2b53584f580310186df7a2055ce3ff83cc0df6caacf1e3489bf
f8cf5d0af5d8
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
      sleep 5
    State:          Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Sun, 17 Apr 2022 18:51:55 +0000
      Finished:     Sun, 17 Apr 2022 18:52:00 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-vnxxr (ro)
Containers:
  green-container-1:
    Container ID:  containerd://97aca76c9a8bcc8613d882e7f464e04182ef4630a6bb5296090e80b34eefa0e3
    Image:         busybox:1.28
    Image ID:      docker.io/library/busybox@sha256:141c253bc4c3fd0a201d32dc1f493bcf3fff003b6df416dea4f4
```

1 2 3 4 5 6 7 8 9

What is the image used by the `initContainer` on the `blue` pod?

busybox

busybox:1.28

nginx

alpine

busybox:1.1


```
controplane ~ ➜  k describe pod blue
Name:         blue
Namespace:    default
Priority:     0
Node:         controlplane/172.25.0.75
Start Time:   Sun, 17 Apr 2022 18:51:51 +0000
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.42.0.11
IPs:
  IP:  10.42.0.11
Init Containers:
  init-myservice:
    Container ID:  containerd://28fa7c4e9d67b048794557903be8a61357eba3fa1e30568dcb3e30aa52adfbcd
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:d2b53584f580310186df7a2055ce3ff83cc0df6caacf1e3489bf
f8cf5d0af5d8
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
      sleep 5
    State:          Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Sun, 17 Apr 2022 18:51:55 +0000
      Finished:     Sun, 17 Apr 2022 18:52:00 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-vnxxr (ro)
Containers:
  green-container-1:
    Container ID:  containerd://97aca76c9a8bcc8613d882e7f464e04182ef4630a6bb5296090e80b34eefa0e3
    Image:         busybox:1.28
    Image ID:      docker.io/library/busybox@sha256:141c253bc4c3fd0a201d32dc1f493bcf3fff003b6df416dea4f4
```

1 2 3 4 5 6 7 8 9

Why is the `initContainer` terminated? What is the reason?

The process crashed

The process completed successfully

The process cannot start

```
controlplane ~ ➜  k describe pod blue
Name:         blue
Namespace:    default
Priority:     0
Node:         controlplane/172.25.0.75
Start Time:   Sun, 17 Apr 2022 18:51:51 +0000
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.42.0.11
IPs:
  IP:  10.42.0.11
Init Containers:
  init-myservice:
    Container ID:  containerd://28fa7c4e9d67b048794557903be8a61357eba3fa1e30568dcb3e30aa52adfbcd
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:d2b53584f580310186df7a2055ce3ff83cc0df6caacf1e3489bf
f8cf5d0af5d8
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
      sleep 5
    State:          Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Sun, 17 Apr 2022 18:51:55 +0000
      Finished:     Sun, 17 Apr 2022 18:52:00 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-vnxxr (ro)
Containers:
  green-container-1:
    Container ID:  containerd://97aca76c9a8bcc8613d882e7f464e04182ef4630a6bb5296090e80b34eefa0e3
    Image:         busybox:1.28
    Image ID:      docker.io/library/busybox@sha256:141c253bc4c3fd0a201d32dc1f493bcf3fff003b6df416dea4f4
```

1 2 3 4 5 6 7 8 9

We just created a new app named `purple`. How many `initContainers` does it have?

1

2

3

4

```
controlplane ~ ➜  k describe pod purple
Name:         purple
Namespace:    default
Priority:     0
Node:         controlplane/172.25.0.75
Start Time:   Sun, 17 Apr 2022 18:55:07 +0000
Labels:       <none>
Annotations:  <none>
Status:       Pending
IP:           10.42.0.12
IPs:
  IP:  10.42.0.12
Init Containers:
  warm-up-1:
    Container ID:  containerd://c4477c75db73f609c7b892d97c8b8d28889262ecb3e43566755c065c306f3594
    Image:         busybox:1.28
    Image ID:      docker.io/library/busybox@sha256:141c253bc4c3fd0a201d32dc1f493bcf3fff003b6df416dea4f4
1046e0f37d47
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
      sleep 600
    State:          Running
      Started:      Sun, 17 Apr 2022 18:55:09 +0000
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-a
  warm-up-2:
    Container ID:
    Image:         busybox:1.28
    Image ID:
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
```

Snipping Tool

Screenshot copied to clipboard and saved

Select here to mark up and share the image

1 2 3 4 5 6 7 8 9

What is the state of the POD?

- Running
- Terminated
- Pending

```
controplane ~ ➜  k get pod purple
NAME     READY   STATUS     RESTARTS   AGE
purple   0/1     Init:0/2   0          11s

controplane ~ ➜  k describe pod purple
Name:         purple
Namespace:    default
Priority:     0
Node:         controlplane/172.25.0.75
Start Time:   Sun, 17 Apr 2022 18:55:07 +0000
Labels:       <none>
Annotations:  <none>
Status:       Pending
IP:           10.42.0.12
IPs:
  IP:  10.42.0.12
Init Containers:
  warm-up-1:
    Container ID:  containerd://c4477c75db73f609c7b892d97c8b8d28889262ecb3e43566755c065c306f3594
    Image:         busybox:1.28
    Image ID:      docker.io/library/busybox@sha256:141c253bc4c3fd0a201d32dc1f493bcf3fff003b6df416dea4f4
1046e0f37d47
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
      sleep 600
    State:          Running
      Started:      Sun, 17 Apr 2022 18:55:09 +0000
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-sltl5 (ro)
  warm-up-2:
    Container ID:
    Image:         busybox:1.28
    Image ID:
```

1 2 3 4 5 6 7 8 9

How long after the creation of the POD will the application come up and be available to users?

- 10 Minutes
- 20 Minutes
- 30 Minutes
- 600 Seconds

```
      Reason:       PodInitializing
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-sltl5 (ro)
Containers:
  purple-container:
    Container ID:
    Image:         busybox:1.28
    Image ID:
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
      echo The app is running! && sleep 3600
    State:          Waiting
      Reason:       PodInitializing
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-sltl5 (ro)
Conditions:
  Type              Status
  Initialized       False
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  kube-api-access-sltl5:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
```

1 2 3 4 5 6 7 8 9

Update the pod `red` to use an `initContainer` that uses the `busybox` image and `sleeps for 20` seconds

Delete and re-create the pod if necessary. But make sure no other configurations change.

Check

- Pod: red
- initContainer Configured Correctly

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2022-04-17T18:51:51Z"
  name: red
  namespace: default
  resourceVersion: "786"
  uid: 4ae14a42-5a0e-4346-8c95-d7a062966291
spec:
  initContainers:
  - image: busybox
    name: busybox
    command: [ "sleep", "20" ]
  containers:
  - command:
    - sh
    - -c
    - echo The app is running! && sleep 3600
    image: busybox:1.28
    imagePullPolicy: IfNotPresent
    name: red-container
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-lcs9s
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: controlplane
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
```

1 2 3 4 5 6 7 8 9

Update the pod `red` to use an `initContainer` that uses the `busybox` image and `sleeps for 20` seconds

Delete and re-create the pod if necessary. But make sure no other configurations change.

Complete Next

- Pod: red
- initContainer Configured Correctly

```
controlplane ~ ➜  k get pod red
NAME   READY   STATUS    RESTARTS   AGE
red    1/1     Running   0          5m52s

controlplane ~ ➜  k edit pod red
error: pods "red" is invalid
A copy of your changes has been stored to "/tmp/kubectl-edit-847706811.yaml"
error: Edit cancelled, no valid changes were saved.

controlplane ~ ✖

controlplane ~ ✖ k replace --force -f /tmp/kubectl-edit-847706811.yaml
pod "red" deleted
pod/red replaced

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9

A new application `orange` is deployed. There is something wrong with it. Identify and fix the issue. Once fixed, wait for the application to run before checking solution.

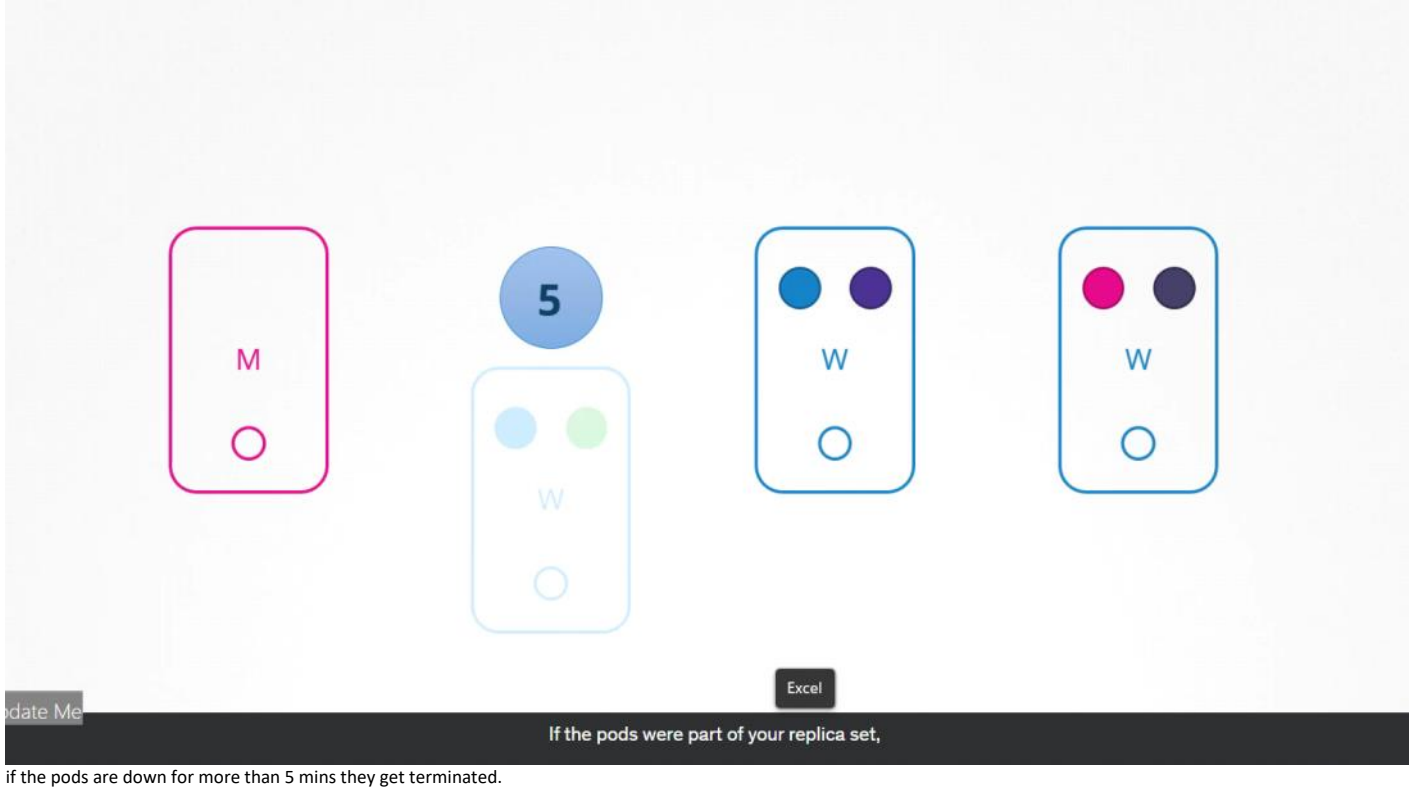
Check

Issue fixed

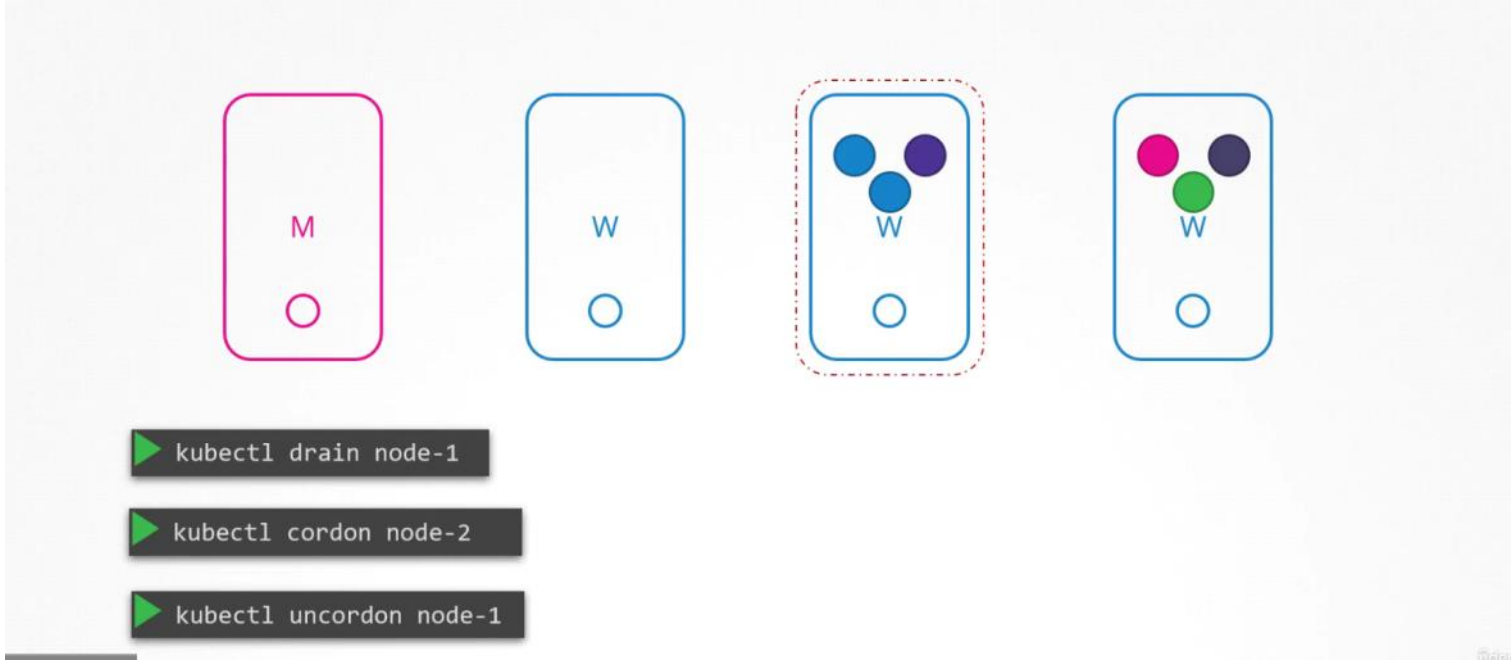
```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2022-04-17T18:59:41Z"
  name: orange
  namespace: default
  resourceVersion: "979"
  uid: c901e803-aa9c-4c43-b6ec-68e897e87357
spec:
  containers:
  - command:
    - sh
    - -c
    - echo The app is running! && sleep 3600
    image: busybox:1.28
    imagePullPolicy: IfNotPresent
    name: orange-container
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-ztghh
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  initContainers:
  - command:
    - sh
    - -c
    - sleeep 2;
    image: busybox
    imagePullPolicy: Always
    name: init-myservice
    resources: {}
"/tmp/kubectl-edit-3642015925.yaml" 142L, 4112B
```

OS Upgrades

Friday, June 28, 2024 10:40 AM



if the pods are down for more than 5 mins they get terminated.
If they are a part of replica set they get created on another pod.
When you drain a node they are terminated from that node and get created on another node.
The node is also called uncordoned or unscheduleable



node has to be uncordon so that pods can be scheduleded on it again
codon marks the node as unscheduleable

kubectl drain node01 --ignore-daemonsets

Practice- OS Upgrade

Friday, June 28, 2024 10:50 AM

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

Let us explore the environment first. How many nodes do you see in the cluster?

Including the controlplane and worker nodes.

```
root@controlplane:~# alias k=kubectl
root@controlplane:~# k get nodes
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   12m   v1.20.0
node01         Ready    <none>                 12m   v1.20.0
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

How many applications do you see hosted on the cluster?

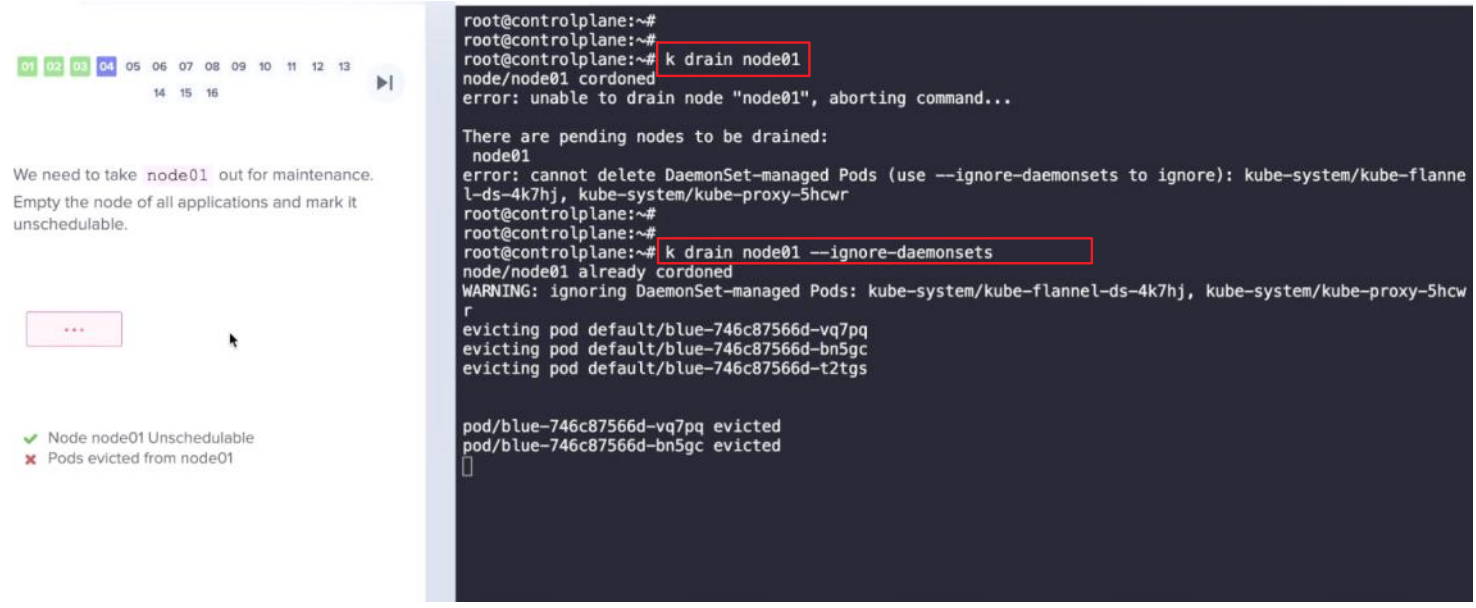
Check the number of deployments.

```
root@controlplane:~# alias k=kubectl
root@controlplane:~# k get nodes
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   12m   v1.20.0
node01         Ready    <none>                 12m   v1.20.0
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get deploy
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
blue   3/3     3            3           7s
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

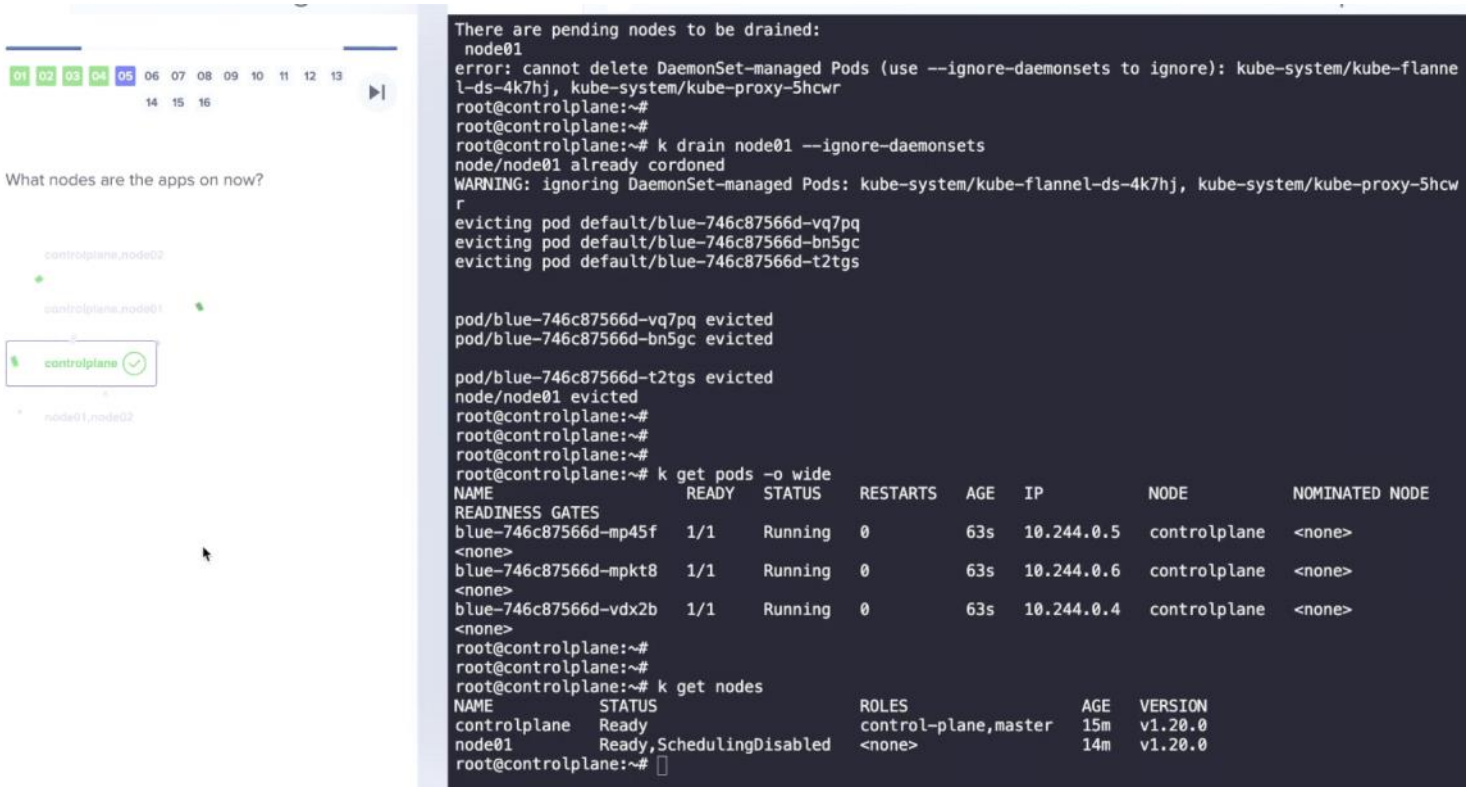
Which nodes are the applications hosted on?

```
root@controlplane:~# alias k=kubectl
root@controlplane:~# k get nodes
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   12m   v1.20.0
node01         Ready    <none>                 12m   v1.20.0
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get deploy
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
blue   3/3     3            3           7s
root@controlplane:~# k get pods
NAME                    READY   STATUS    RESTARTS   AGE
blue-746c87566d-bn5gc   1/1     Running   0          22s
blue-746c87566d-t2tgs   1/1     Running   0          22s
blue-746c87566d-vq7pq   1/1     Running   0          22s
root@controlplane:~# k get pods -o wide
NAME                    READY   STATUS    RESTARTS   AGE   IP          NODE    NOMINATED NODE   READIN
ESS GATES
blue-746c87566d-bn5gc   1/1     Running   0          27s   10.244.1.3  node01  <none>           <none>
blue-746c87566d-t2tgs   1/1     Running   0          27s   10.244.1.4  node01  <none>           <none>
blue-746c87566d-vq7pq   1/1     Running   0          27s   10.244.1.2  node01  <none>           <none>
root@controlplane:~#
```



kubectl drain node01 --ignore-daemonsets

Running the uncordon command on a node will not automatically schedule pods on the node. When new pods are created, they will be placed on node01.



01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

The maintenance tasks have been completed. Configure the node `node01` to be schedulable again.

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get nodes
NAME           STATUS                     ROLES                  AGE   VERSION
controlplane   Ready                      control-plane,master   15m   v1.20.0
node01         Ready,SchedulingDisabled   <none>                 15m   v1.20.0
root@controlplane:~#
root@controlplane:~# k uncordon node01
node/node01 uncordoned
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get nodes
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   15m   v1.20.0
node01         Ready    <none>                 15m   v1.20.0
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

How many pods are scheduled on `node01` now?

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get nodes
NAME           STATUS                     ROLES                  AGE   VERSION
controlplane   Ready                      control-plane,master   15m   v1.20.0
node01         Ready,SchedulingDisabled   <none>                 15m   v1.20.0
root@controlplane:~#
root@controlplane:~# k uncordon node01
node/node01 uncordoned
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get nodes
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   15m   v1.20.0
node01         Ready    <none>                 15m   v1.20.0
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE     IP           NODE           NOMINATED NODE
  READINESS GATES
blue-746c87566d-mp45f    1/1     Running   0          2m14s   10.244.0.5   controlplane   <none>
  <none>
blue-746c87566d-mpkt8    1/1     Running   0          2m14s   10.244.0.6   controlplane   <none>
  <none>
blue-746c87566d-vdx2b    1/1     Running   0          2m14s   10.244.0.4   controlplane   <none>
  <none>
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

Why are the pods placed on the `controlplane` node?

Check the controlplane node details.

- controlplane node does not have any taints
- controlplane node is faulty
- you can never have pods on master nodes
- controlplane node has taints set on it
- controlplane node is cordoned

```
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=controlplane
                    kubernetes.io/os=linux
                    node-role.kubernetes.io/control-plane=
                    node-role.kubernetes.io/master=
Annotations:        flannel.alpha.coreos.com/backend-data: {"VNI":1,"VtepMAC":"1e:4c:dc:69:ad:e5"}
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 10.42.115.9
                    kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Sun, 17 Apr 2022 19:02:46 +0000
Taints:             <none>
Unschedulable:      false
Lease:
  HolderIdentity:  controlplane
  AcquireTime:     <unset>
  RenewTime:       Sun, 17 Apr 2022 19:20:26 +0000
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime                Reaso
n                       Message
  ----                 ------  -----------------                 ------------------                -----
-                       -------
  NetworkUnavailable   False   Sun, 17 Apr 2022 19:03:13 +0000   Sun, 17 Apr 2022 19:03:13 +0000   Flann
elIsUp                  Flannel is running on this node
  MemoryPressure       False   Sun, 17 Apr 2022 19:18:23 +0000   Sun, 17 Apr 2022 19:02:36 +0000   Kubel
etHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         False   Sun, 17 Apr 2022 19:18:23 +0000   Sun, 17 Apr 2022 19:02:36 +0000   Kubel
etHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure          False   Sun, 17 Apr 2022 19:18:23 +0000   Sun, 17 Apr 2022 19:02:36 +0000   Kubel
etHasSufficientPID      kubelet has sufficient PID available
  Ready                True    Sun, 17 Apr 2022 19:18:23 +0000   Sun, 17 Apr 2022 19:03:16 +0000   Kubel
etReady                 kubelet is posting ready status
Addresses:
  InternalIP:  10.42.115.9
  Hostname:    controlplane
Capacity:
```

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

What is the name of the POD hosted on `node01` that is not part of a replicaset?

- simple-webapp-1
- red
- redis
- hr-app
- blue

```
root@controlplane:~#
root@controlplane:~# kubectl drain node01 --ignore-daemonsets
node/node01 cordoned
error: unable to drain node "node01", aborting command...

There are pending nodes to be drained:
 node01
error: cannot delete Pods not managed by ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSe
t (use --force to override): default/hr-app
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get pods
NAME                    READY   STATUS    RESTARTS   AGE
blue-746c87566d-mp45f   1/1     Running   0          5m17s
blue-746c87566d-mpkt8   1/1     Running   0          5m17s
blue-746c87566d-vdx2b   1/1     Running   0          5m17s
hr-app                  1/1     Running   0          61s
root@controlplane:~# k get pods -o wide
NAME                    READY   STATUS    RESTARTS   AGE     IP           NODE           NOMINATED NODE
  READINESS GATES
blue-746c87566d-mp45f   1/1     Running   0          5m34s   10.244.0.5   controlplane   <none>
  <none>
blue-746c87566d-mpkt8   1/1     Running   0          5m34s   10.244.0.6   controlplane   <none>
  <none>
blue-746c87566d-vdx2b   1/1     Running   0          5m34s   10.244.0.4   controlplane   <none>
  <none>
hr-app                  1/1     Running   0          78s     10.244.1.5   node01         <none>
  <none>
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

What would happen to `hr-app` if `node01` is drained forcefully?

Try it and see for yourself.

- hr-app will be lost forever
- hr-app will continue to run as a Docker container
- hr-app will be recreated on other nodes
- hr-app will be re-created on master

```
root@controlplane:~#
root@controlplane:~# kubectl drain node01 --ignore-daemonsets
node/node01 cordoned
error: unable to drain node "node01", aborting command...

There are pending nodes to be drained:
 node01
error: cannot delete Pods not managed by ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSe
t (use --force to override): default/hr-app
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get pods
NAME                    READY   STATUS    RESTARTS   AGE
blue-746c87566d-mp45f   1/1     Running   0          5m17s
blue-746c87566d-mpkt8   1/1     Running   0          5m17s
blue-746c87566d-vdx2b   1/1     Running   0          5m17s
hr-app                  1/1     Running   0          61s
root@controlplane:~# k get pods -o wide
NAME                    READY   STATUS    RESTARTS   AGE     IP           NODE           NOMINATED NODE
  READINESS GATES
blue-746c87566d-mp45f   1/1     Running   0          5m34s   10.244.0.5   controlplane   <none>
  <none>
blue-746c87566d-mpkt8   1/1     Running   0          5m34s   10.244.0.6   controlplane   <none>
  <none>
blue-746c87566d-vdx2b   1/1     Running   0          5m34s   10.244.0.4   controlplane   <none>
  <none>
hr-app                  1/1     Running   0          78s     10.244.1.5   node01         <none>
  <none>
root@controlplane:~# kubectl drain node01 --ignore-daemonsets --force
node/node01 already cordoned
WARNING: deleting Pods not managed by ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet:
default/hr-app; ignoring DaemonSet-managed Pods: kube-system/kube-flannel-ds-4k7hj, kube-system/kube-pro
xy-5hcwr
evicting pod default/hr-app
```

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

Oops! We did not want to do that! `hr-app` is a critical application that should not be destroyed. We have now reverted back to the previous state and re-deployed `hr-app` as a deployment.

Ok

```
root@controlplane:~# k get deploy
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
blue     3/3     3            3           9m36s
hr-app   1/1     1            1           17s
root@controlplane:~# k get pods -o wide
NAME                      READY   STATUS    RESTARTS   AGE     IP           NODE           NOMINATED NOD
E   READINESS GATES
blue-746c87566d-mp45f     1/1     Running   0          8m29s   10.244.0.5   controlplane   <none>
    <none>
blue-746c87566d-mpkt8     1/1     Running   0          8m29s   10.244.0.6   controlplane   <none>
    <none>
blue-746c87566d-vdx2b     1/1     Running   0          8m29s   10.244.0.4   controlplane   <none>
    <none>
hr-app-76d475c57d-9vmts   1/1     Running   0          24s     10.244.1.6   node01         <none>
    <none>
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

`hr-app` is a critical app and we do not want it to be removed and we do not want to schedule any more pods on `node01`.

Mark `node01` as `unschedulable` so that no new pods are scheduled on this node.

Make sure that `hr-app` is not affected.

Check

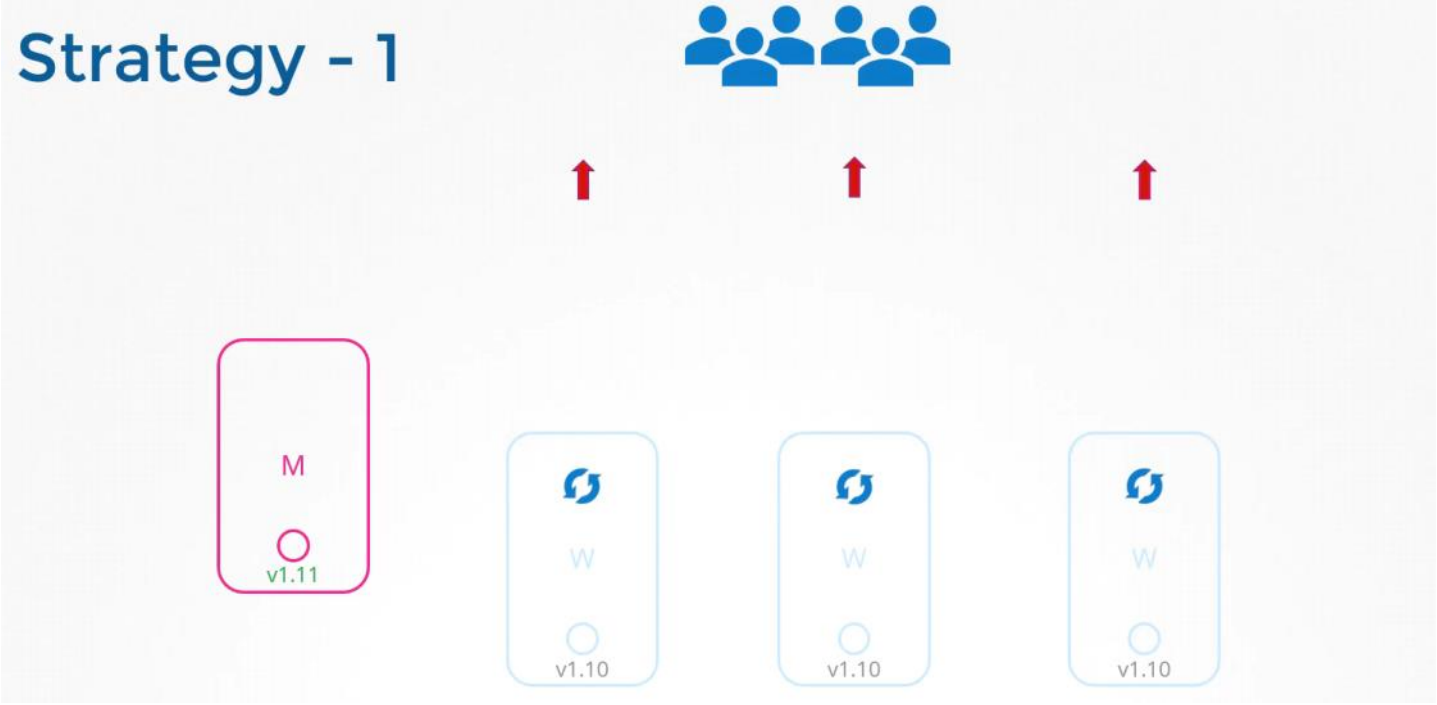
- Node01 Unschedulable
- hr-app still running on node01?

```
root@controlplane:~# k get deploy
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
blue     3/3     3            3           9m36s
hr-app   1/1     1            1           17s
root@controlplane:~# k get pods -o wide
NAME                      READY   STATUS    RESTARTS   AGE     IP           NODE           NOMINATED NOD
E   READINESS GATES
blue-746c87566d-mp45f     1/1     Running   0          8m29s   10.244.0.5   controlplane   <none>
    <none>
blue-746c87566d-mpkt8     1/1     Running   0          8m29s   10.244.0.6   controlplane   <none>
    <none>
blue-746c87566d-vdx2b     1/1     Running   0          8m29s   10.244.0.4   controlplane   <none>
    <none>
hr-app-76d475c57d-9vmts   1/1     Running   0          24s     10.244.1.6   node01         <none>
    <none>
root@controlplane:~# k get nodes
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   23m   v1.20.0
node01         Ready    <none>                 22m   v1.20.0
root@controlplane:~# k cordon node01
node/node01 cordoned
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get nodes
NAME           STATUS                     ROLES                  AGE   VERSION
controlplane   Ready                      control-plane,master   23m   v1.20.0
node01         Ready,SchedulingDisabled   <none>                 23m   v1.20.0
root@controlplane:~#
```

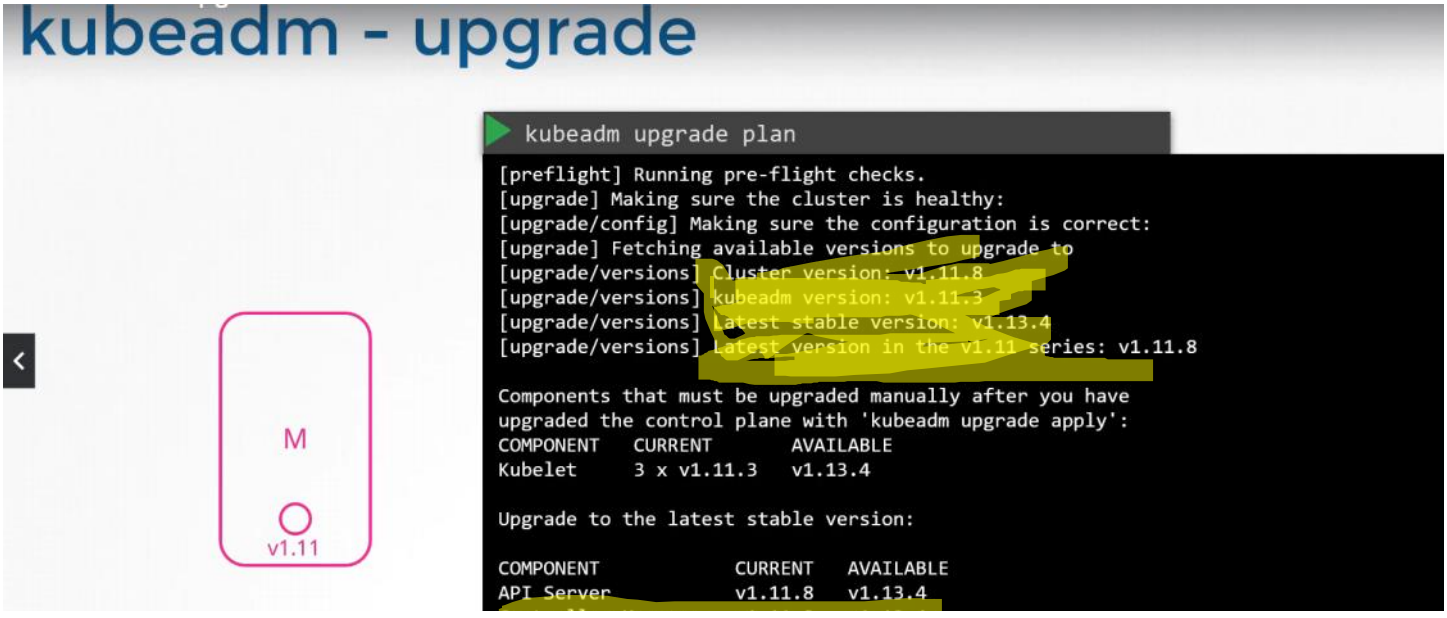
kubernetes software versions

Thursday, August 22, 2024 5:27 PM

no other services can be at a higher verson than kube api server
kuberneets support the latest three minor versions only
you must upgrade one minor version at a time
if the master node is being upgraded the aplication will still be running bbut no new pods can be scheduled, you cannot reach nodes, other sevices cannot communicate



strategy 1 -- upgrade all worker nodes at a time
strategy 2 -- upgrade few at a time
strategy 3 -- create new nodes with upgraded version and later shift the workload to new nodes



v1.11

```
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
```

Kubeadm upgrade plan

```
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
Etcd                 3.2.18    N/A

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

        kubeadm upgrade apply v1.13.4

Note: Before you can perform this upgrade, you have to update kubeadm to v1.13.4.
```

kubeadm - upgrade

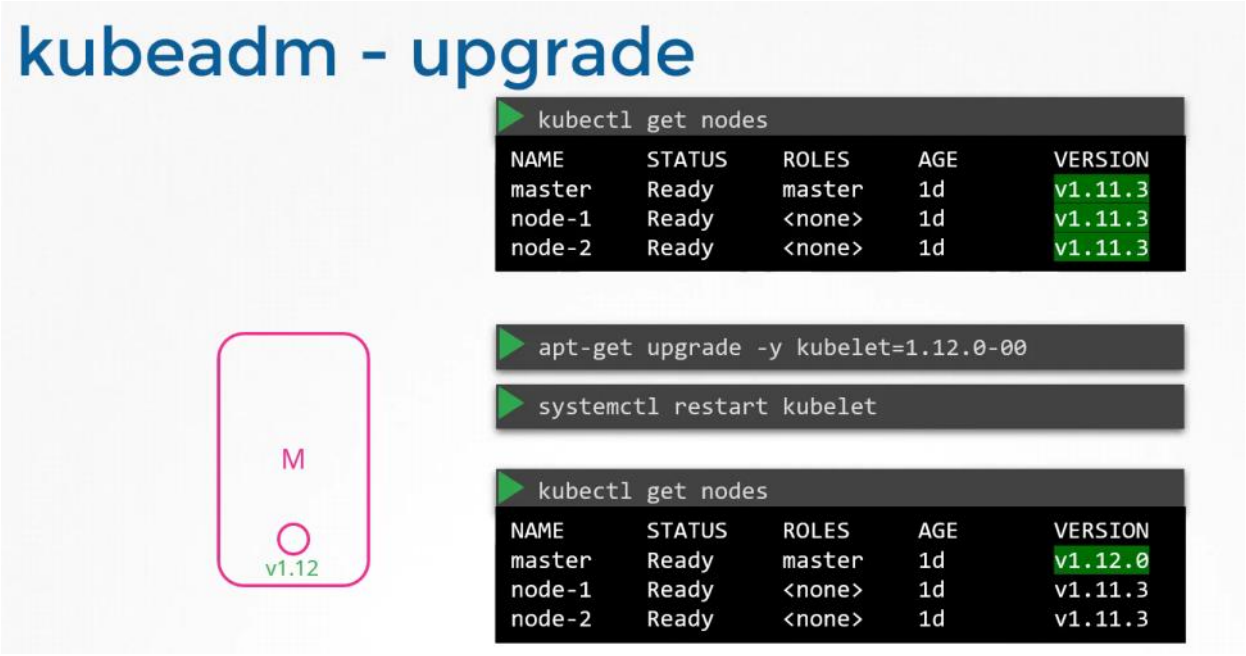
```
apt-get upgrade -y kubeadm=1.12.0-00
```

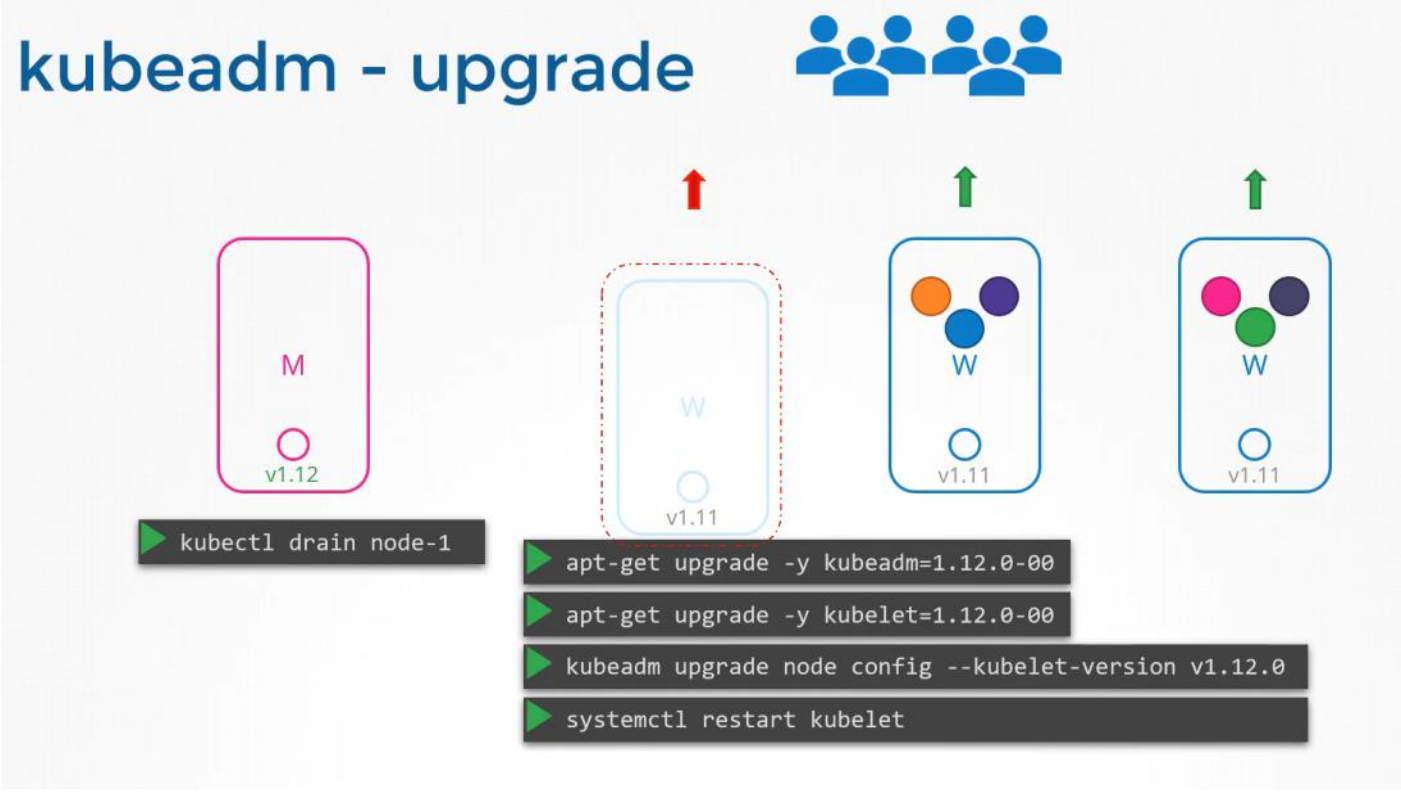
```
kubeadm upgrade apply v1.12.0
…

[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.
```

M

v1.12

```
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
```





after that you will have to mark the node as scheduleable

```
kubectl uncordon node-1
```

On the controlplane node:

Use any text editor you prefer to open the file that defines the Kubernetes apt repository.

```
vim /etc/apt/sources.list.d/kubernetes.list
```

Update the version in the URL to the next available minor release, i.e v1.30.

```
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
```

After making changes, save the file and exit from your text editor. Proceed with the next instruction.

```
apt update
apt-cache madison kubeadm
```

Based on the version information displayed by `apt-cache madison`, it indicates that for Kubernetes version `1.30.0`, the available package version is `1.30.0-1.1`. Therefore, to install kubeadm for Kubernetes `v1.30.0`, use the following command:

```
apt-get install kubeadm=1.30.0-1.1
```

Run the following command to upgrade the Kubernetes cluster.

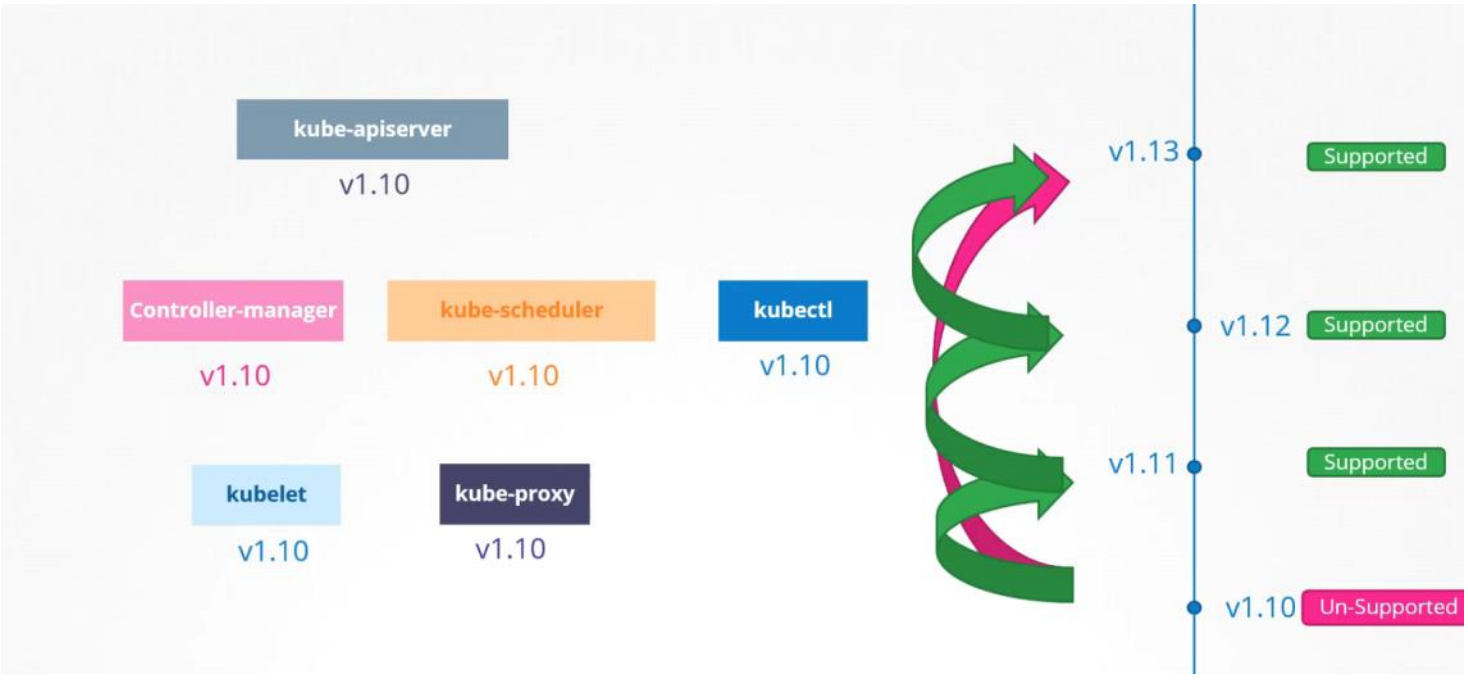
```
kubeadm upgrade plan v1.30.0
kubeadm upgrade apply v1.30.0
```

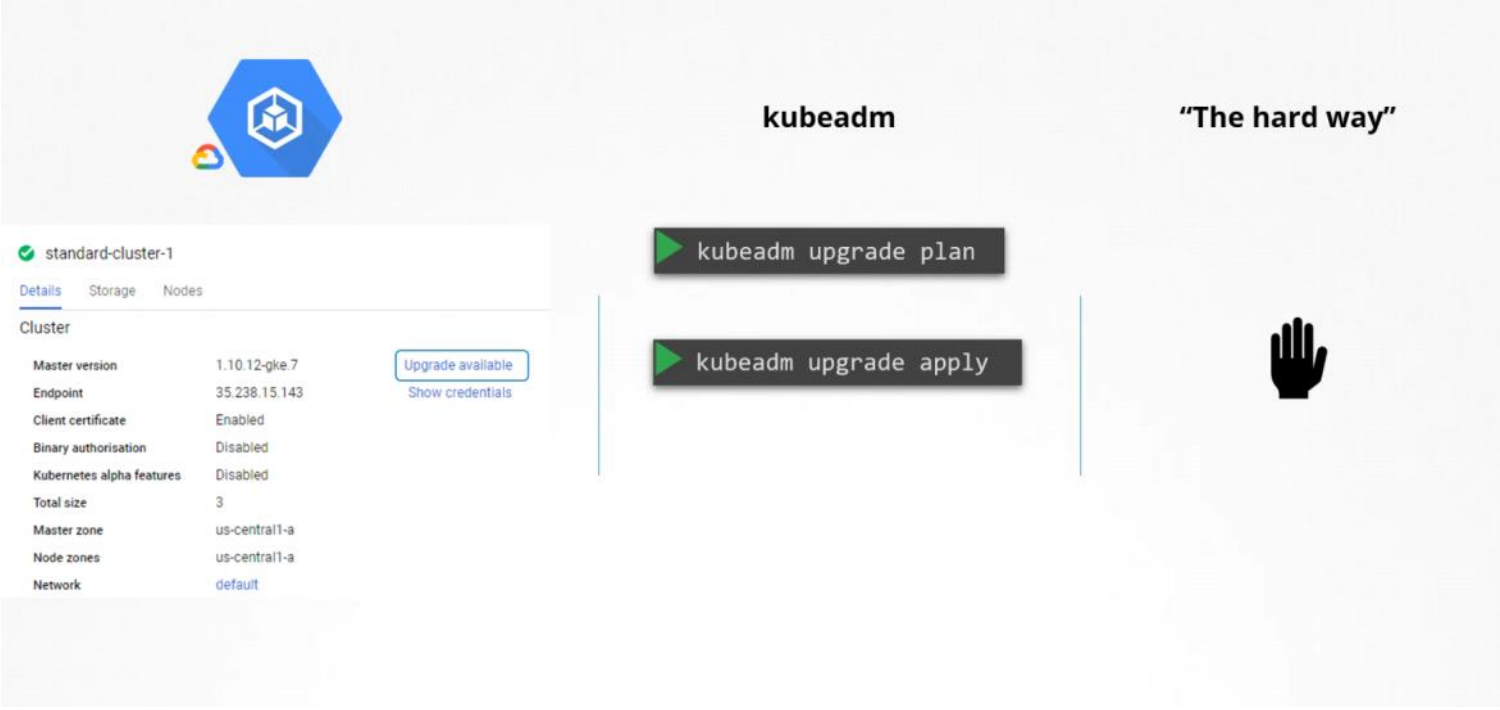
Note that the above steps can take a few minutes to complete.
Now, upgrade the version and restart Kubelet. Also, mark the node (in this case, the "controlplane" node) as schedulable.

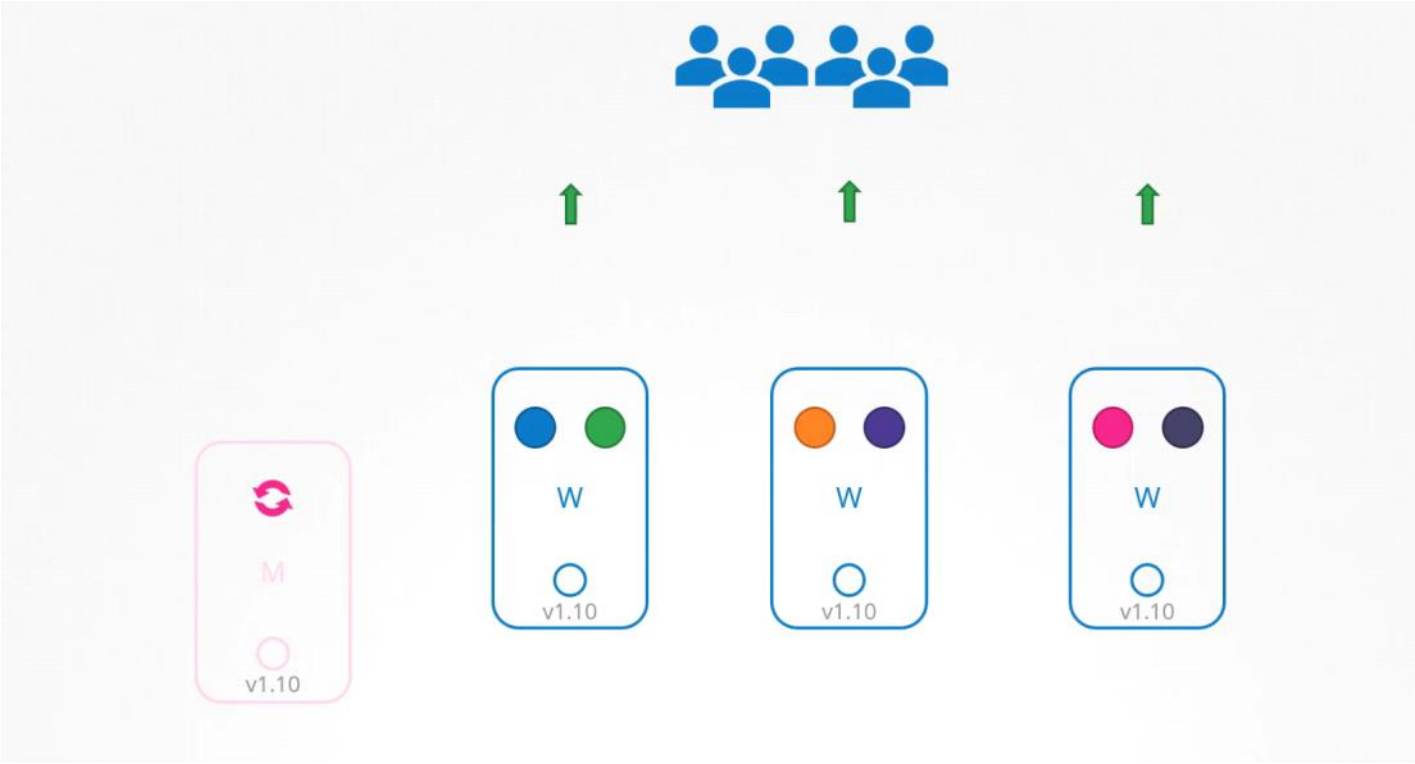
```
apt-get install kubelet=1.30.0-1.1
systemctl daemon-reload
systemctl restart kubelet
```

Cluster Upgrade Process

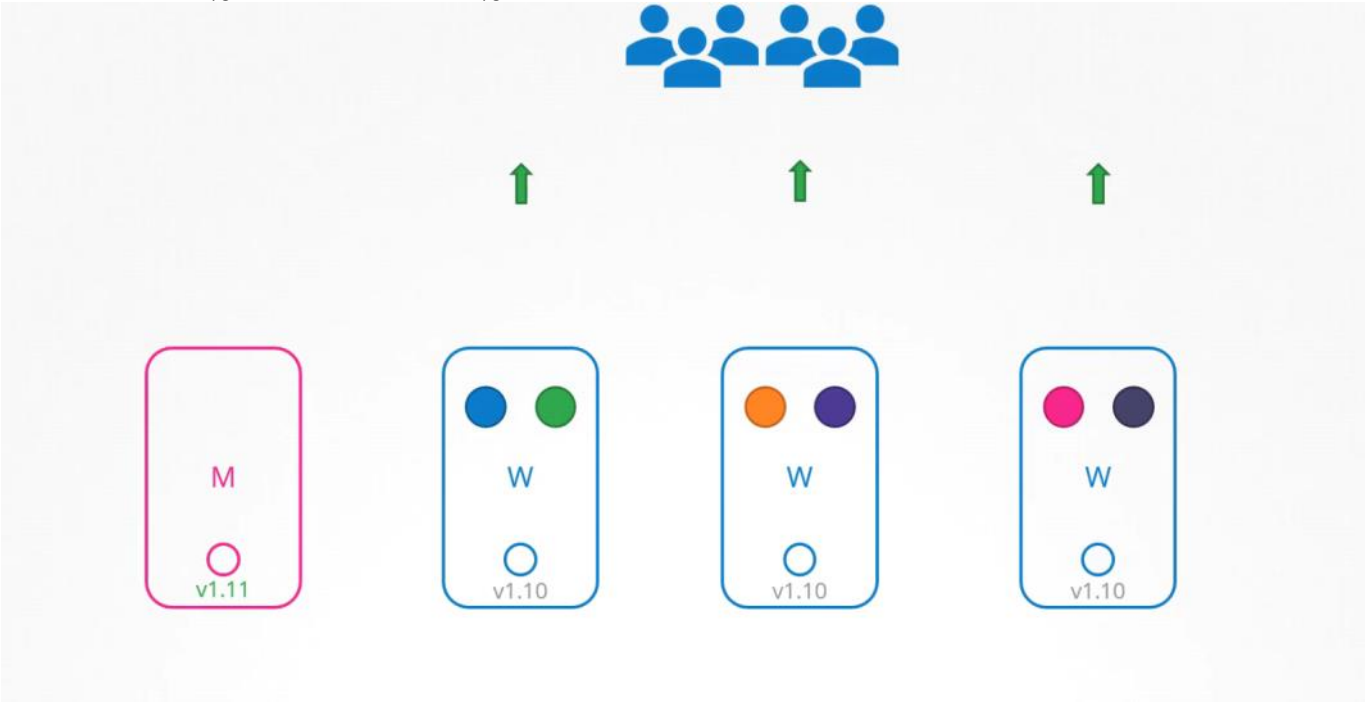
Friday, June 28, 2024 11:23 AM

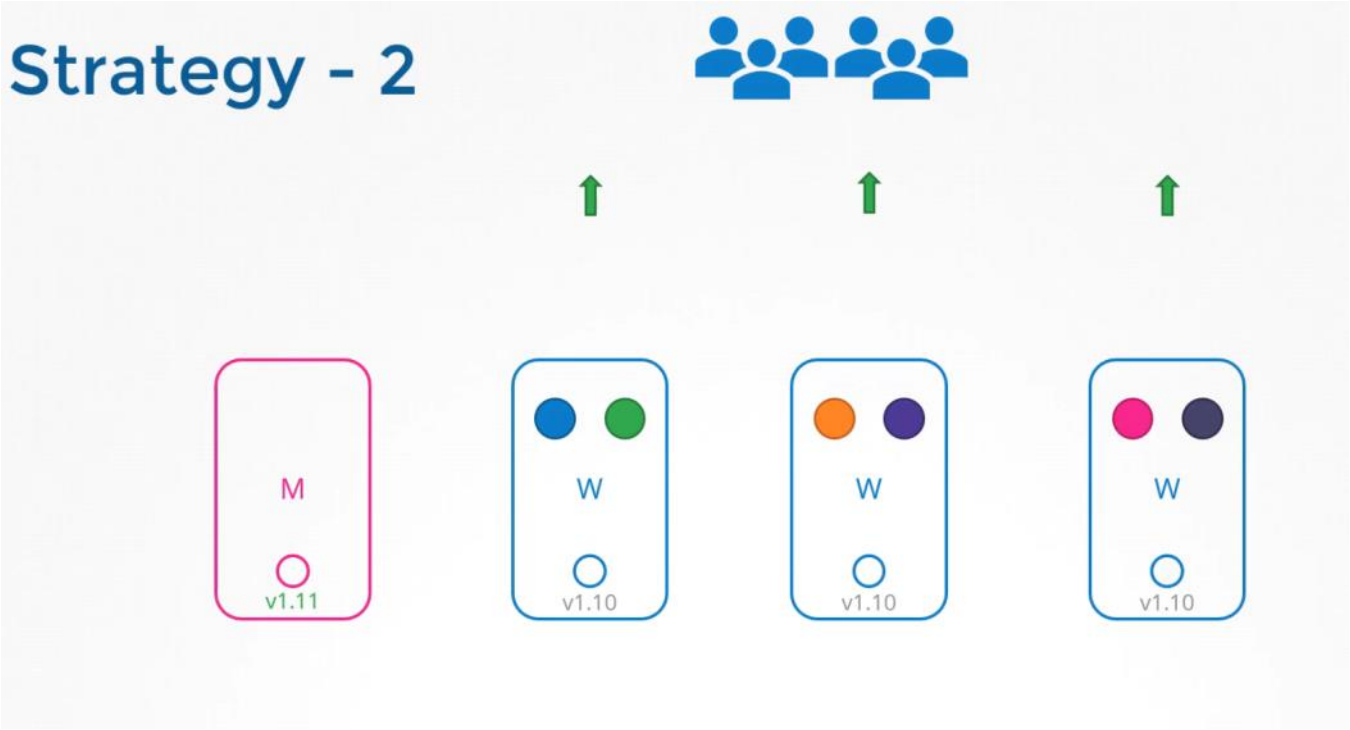




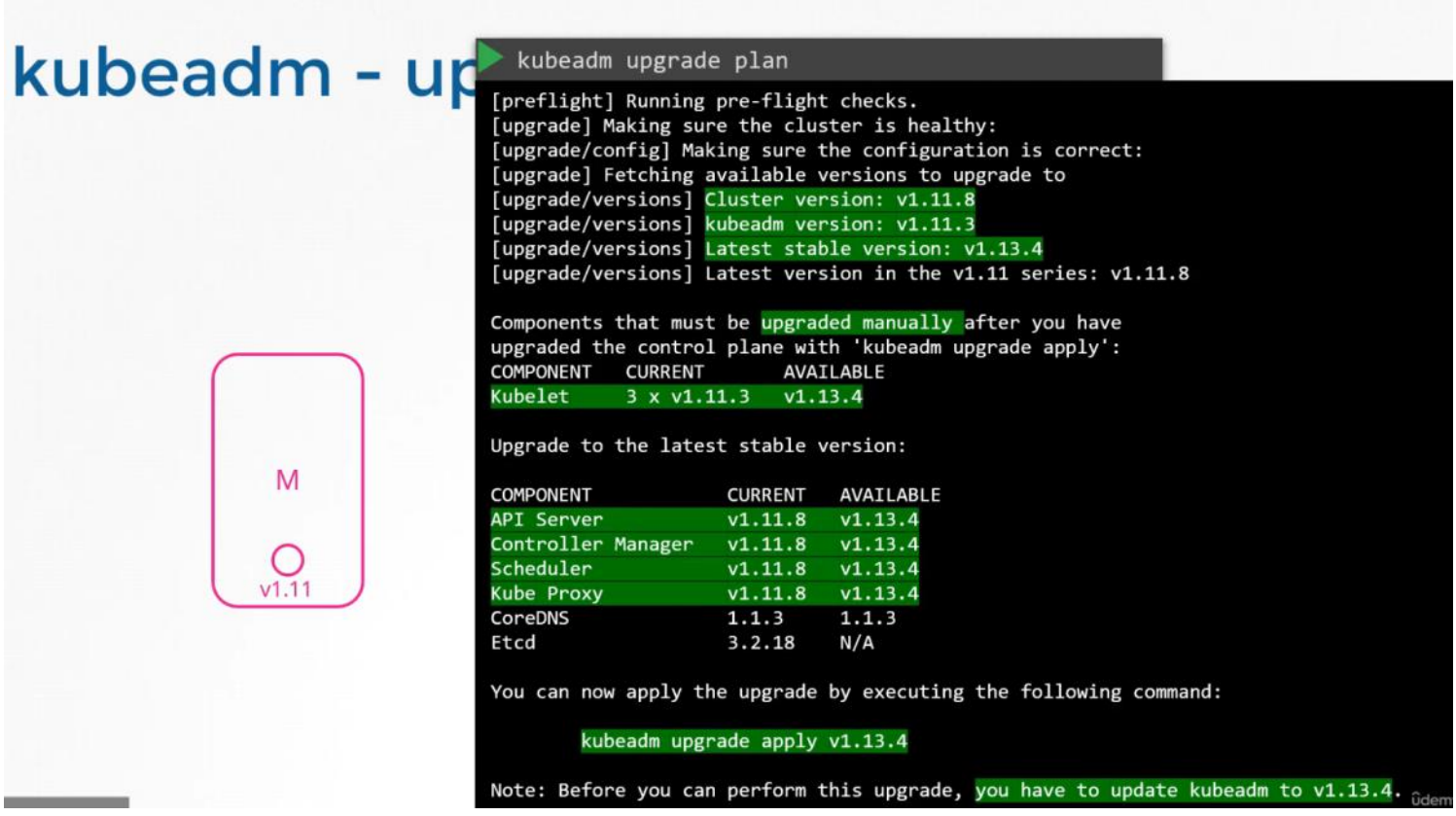


First the master node will upgrade then the worker node will upgrade.





Worker node should be upgraded one at a time.



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.
```

M

v1.12

```
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
```

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
```

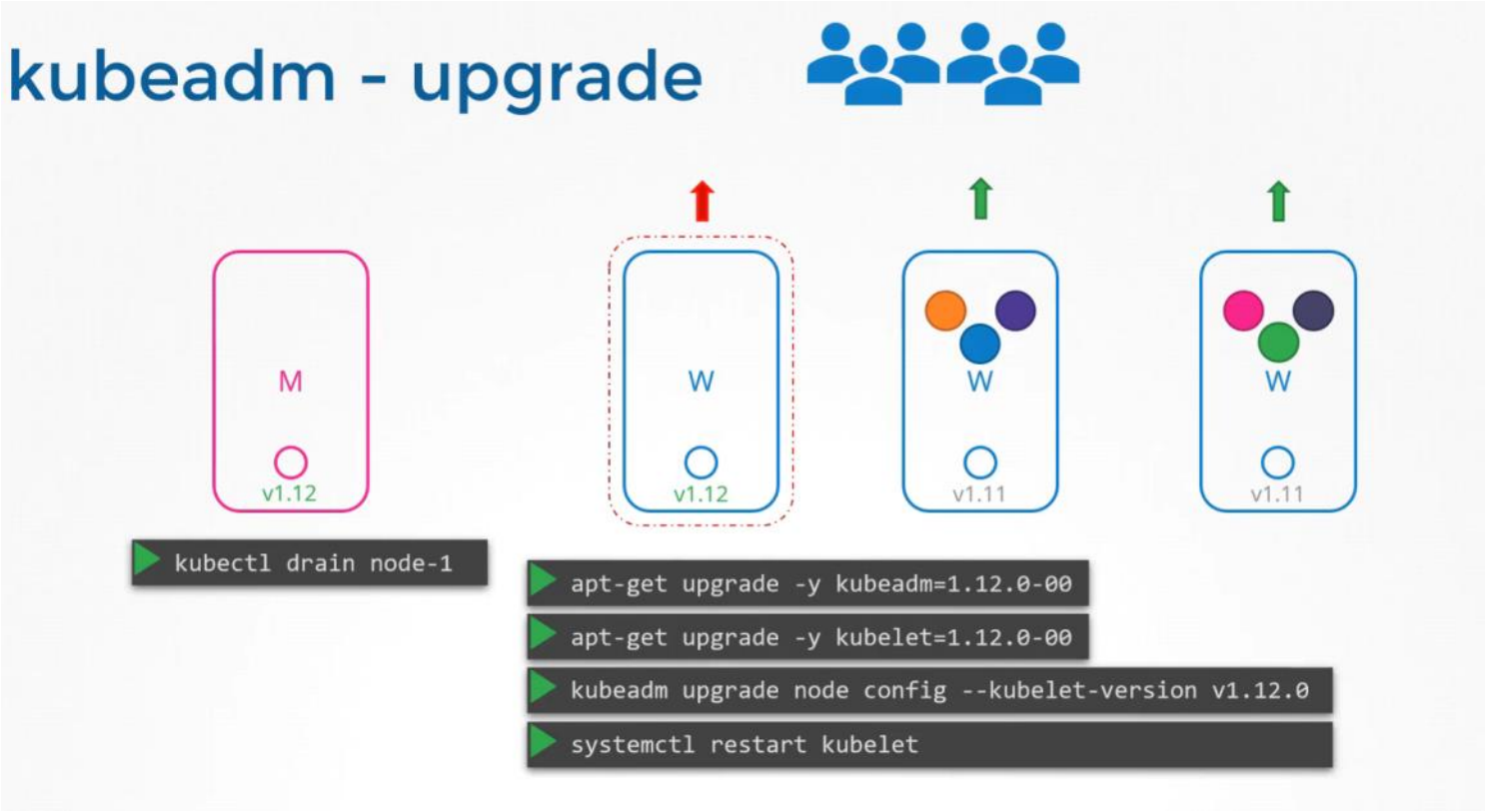
M

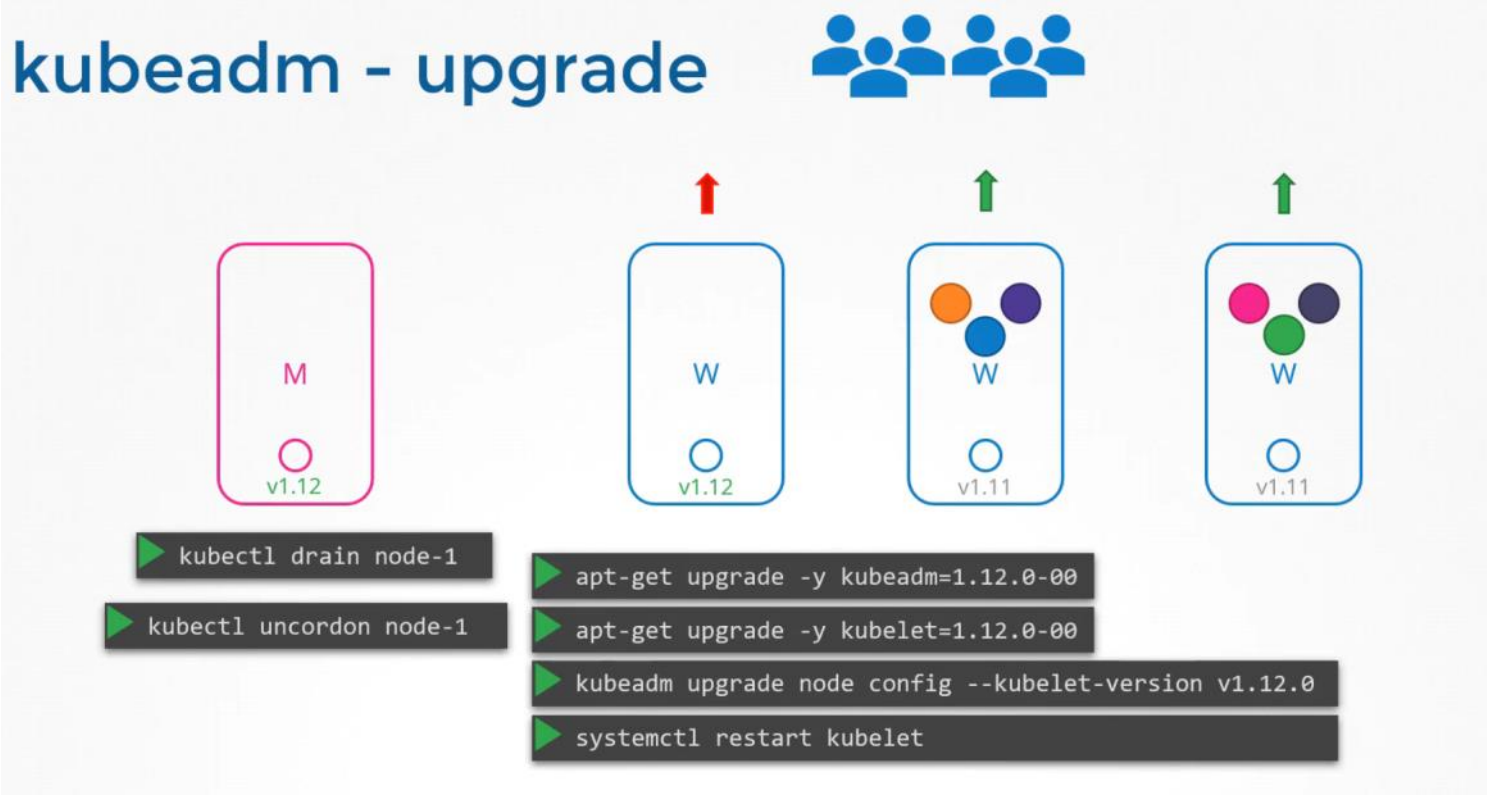
v1.12

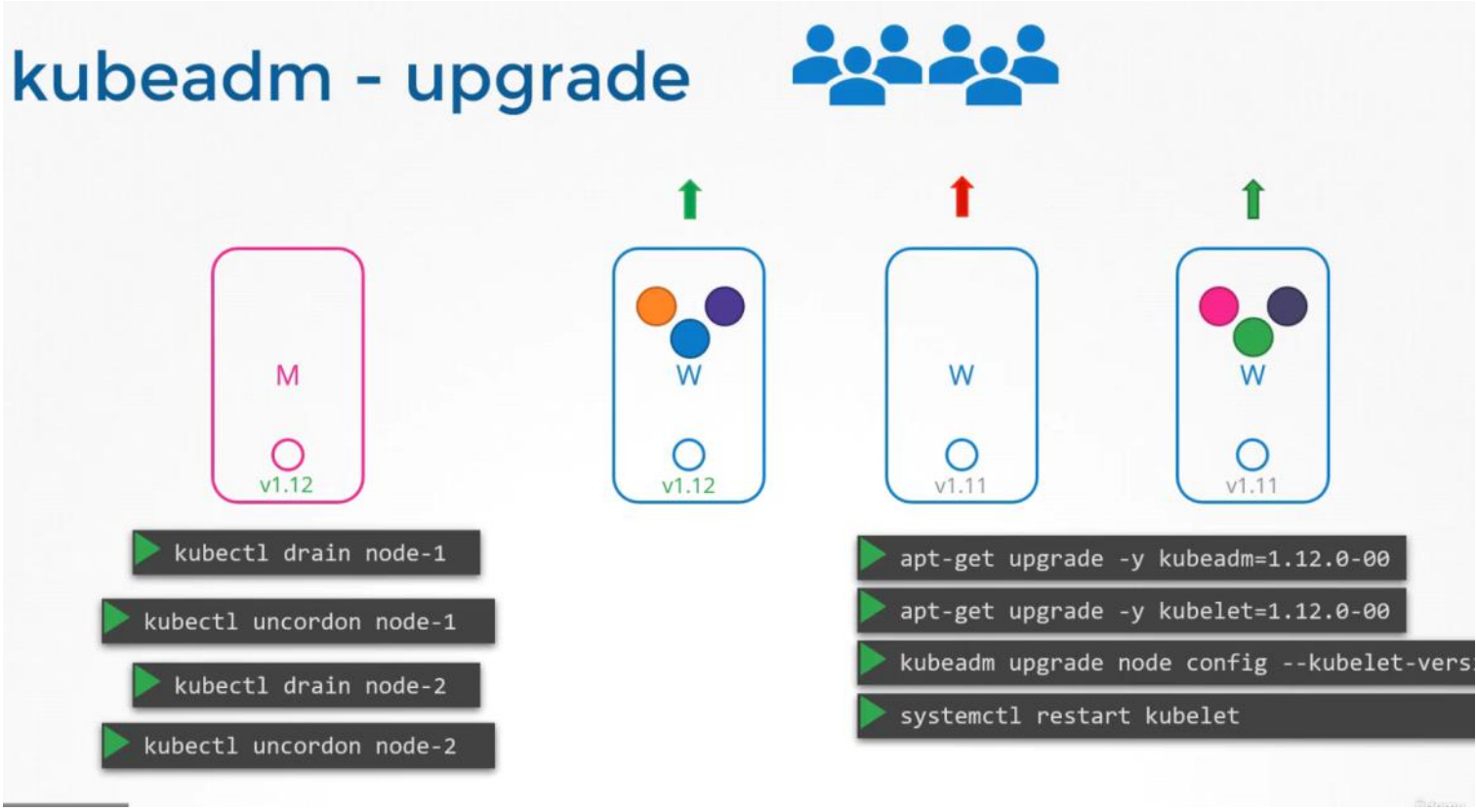
```
apt-get upgrade -y kubelet=1.12.0-00
```

```
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
```

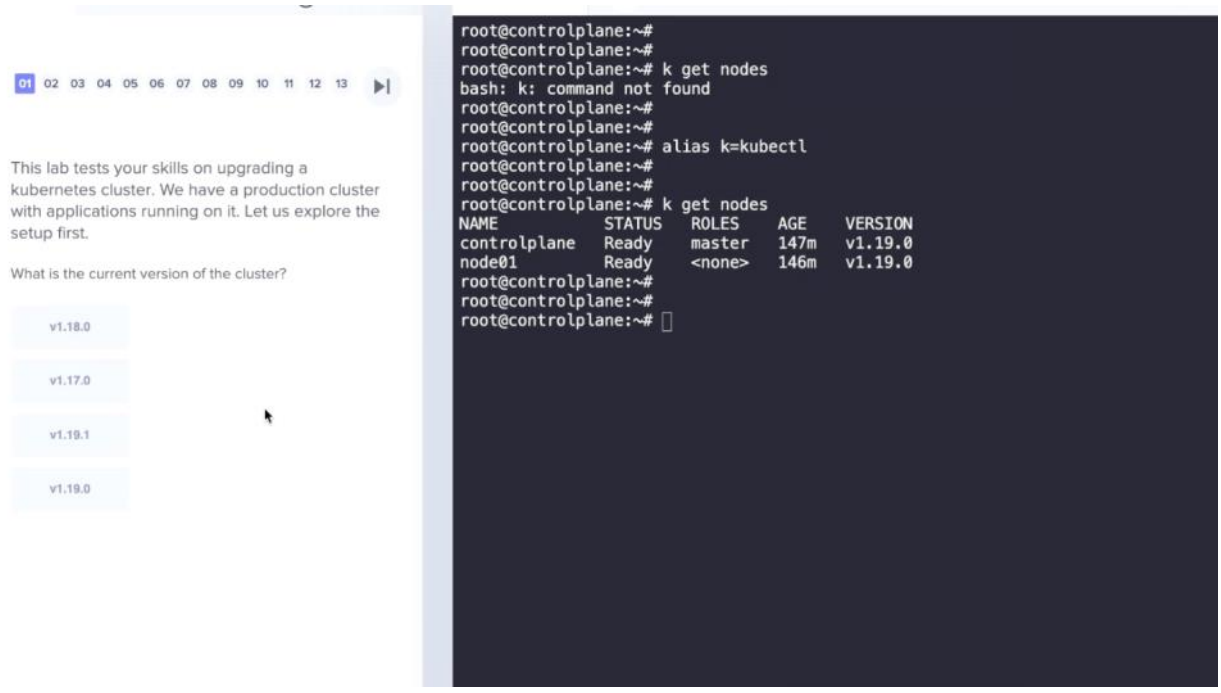

kubeadm - upgrade
M
v1.12
W
v1.12
W
v1.11
W
v1.11
kubectl drain node-1
apt-get upgrade -y kubeadm=1.12.0-00
apt-get upgrade -y kubelet=1.12.0-00
kubeadm upgrade node config --kubelet-version v1.12.0
systemctl restart kubelet

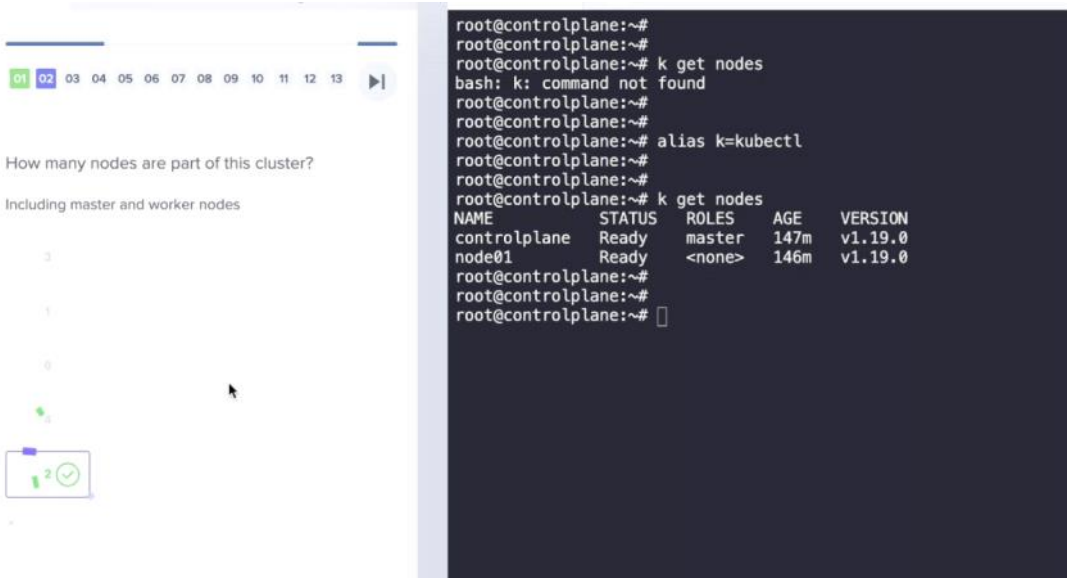

kubeadm - upgrade
M
v1.12
W
v1.12
W
v1.11
W
v1.11
kubectl drain node-1
kubectl uncordon node-1
apt-get upgrade -y kubeadm=1.12.0-00
apt-get upgrade -y kubelet=1.12.0-00
kubeadm upgrade node config --kubelet-version v1.12.0
systemctl restart kubelet

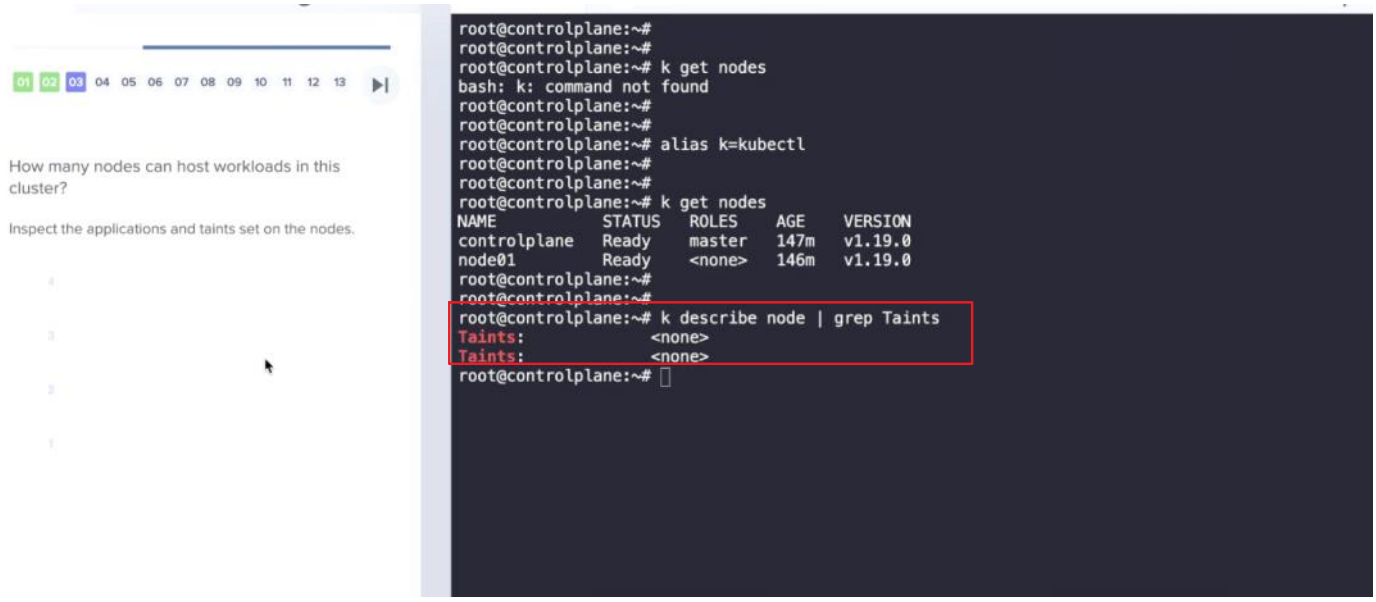

kubeadm - upgrade
M
v1.12
W
v1.12
W
v1.11
W
v1.11
kubectl drain node-1
kubectl uncordon node-1
kubectl drain node-2
kubectl uncordon node-2
apt-get upgrade -y kubeadm=1.12.0-00
apt-get upgrade -y kubelet=1.12.0-00
kubeadm upgrade node config --kubelet-vers
systemctl restart kubelet

Cluster Upgrade

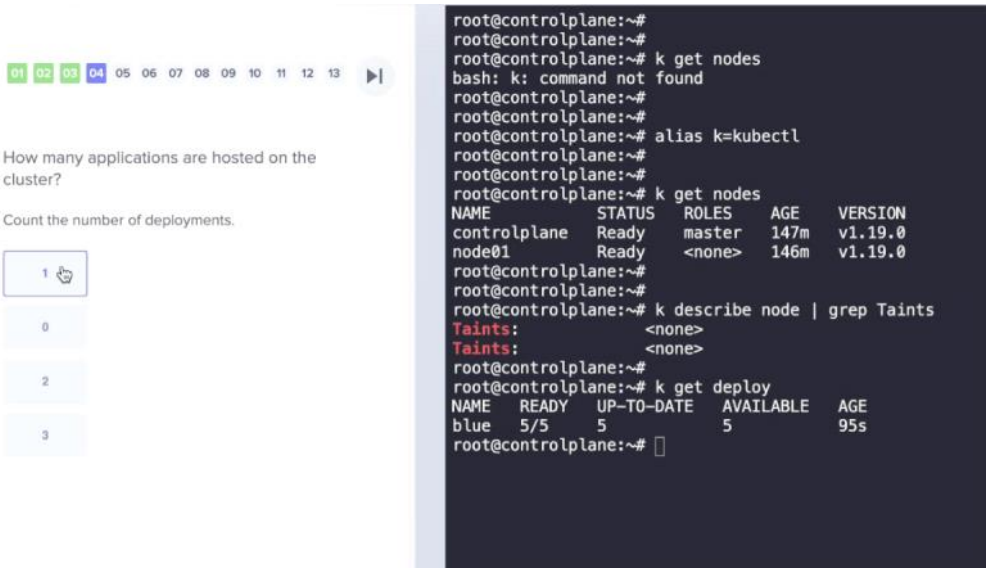
Friday, June 28, 2024 11:37 AM







There are no taints so both the nodes can hos the applications



01 02 03 04 05 06 07 08 09 10 11 12 13

What nodes are the pods hosted on?

- node02
- node01,node02
- master
- node01
- controlplane,node01

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get nodes
bash: k: command not found
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# alias k=kubectl
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get nodes
NAME           STATUS   ROLES    AGE    VERSION
controlplane   Ready    master   147m   v1.19.0
node01         Ready    <none>   146m   v1.19.0
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k describe node | grep Taints
Taints:             <none>
Taints:             <none>
root@controlplane:~#
root@controlplane:~# k get deploy
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
blue   5/5     5            5           95s
root@controlplane:~# k get pods -o wide
NAME                    READY   STATUS    RESTARTS   AGE    IP           NODE           NOMINATED NODE
 READINESS GATES
blue-746c87566d-2c8rs   1/1     Running   0          113s   10.244.0.4   controlplane   <none>
 <none>
blue-746c87566d-4x2bh   1/1     Running   0          113s   10.244.1.4   node01         <none>
 <none>
blue-746c87566d-mvjls   1/1     Running   0          113s   10.244.1.2   node01         <none>
 <none>
blue-746c87566d-ptxxp   1/1     Running   0          113s   10.244.1.3   node01         <none>
 <none>
blue-746c87566d-r8h27   1/1     Running   0          113s   10.244.0.5   controlplane   <none>
 <none>
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12 13

You are tasked to upgrade the cluster. User's accessing the applications must not be impacted. And you cannot provision new VMs. What strategy would you use to upgrade the cluster?

- Upgrade one node at a time while moving the workloads to the other
- Users will be impacted since there is only one worker node
- Upgrade all nodes at once
- Add new nodes with newer versions while taking down existing nodes

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12 13

What is the latest stable version available for upgrade?

Use the `kubeadm` tool

v1.19.16

v1.10.0

v1.13.0

v1.12.1

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubeadm upgrade plan
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-confi
g -oyaml'
[preflight] Running pre-flight checks.
[upgrade] Running cluster health checks
```

01 02 03 04 05 06 07 08 09 10 11 12 13

What is the latest stable version available for upgrade?

Use the `kubeadm` tool

v1.19.16

v1.10.0

v1.13.0

v1.12.1

```
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-confi
g -oyaml'
[preflight] Running pre-flight checks.
[upgrade] Running cluster health checks
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: v1.19.0
[upgrade/versions] kubeadm version: v1.19.0
I0417 19:52:57.728305   36310 version.go:252] remote version is much newer: v1.23.5; falling back to: st
able-1.19
[upgrade/versions] Latest stable version: v1.19.16
[upgrade/versions] Latest stable version: v1.19.16
[upgrade/versions] Latest version in the v1.19 series: v1.19.16
[upgrade/versions] Latest version in the v1.19 series: v1.19.16

Components that must be upgraded manually after you have upgraded the control plane with 'kubeadm upgrad
e apply':
COMPONENT   CURRENT       AVAILABLE
kubelet     2 x v1.19.0   v1.19.16

Upgrade to the latest version in the v1.19 series:

COMPONENT                 CURRENT   AVAILABLE
kube-apiserver            v1.19.0   v1.19.16
kube-controller-manager   v1.19.0   v1.19.16
kube-scheduler            v1.19.0   v1.19.16
kube-proxy                v1.19.0   v1.19.16
CoreDNS                   1.7.0     1.7.0
etcd                      3.4.9-1   3.4.9-1

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

        kubeadm upgrade apply v1.19.16

Note: Before you can perform this upgrade, you have to update kubeadm to v1.19.16.

_____________________________________________________________________
```

01 02 03 04 05 06 07 08 09 10 11 12 13

We will be upgrading the master node first. Drain the master node of workloads and mark it `UnSchedulable`

Check

- Master Node: SchedulingDisabled

```
root@controlplane:~# k drain controlplane
node/controlplane cordoned
error: unable to drain node "controlplane", aborting command...

There are pending nodes to be drained:
 controlplane
error: cannot delete DaemonSet-managed Pods (use --ignore-daemonsets to ignore): kube-system/kube-flanne
l-ds-9q9h4, kube-system/kube-proxy-vfd7p
root@controlplane:~# k drain controlplane --ignore-daemonsets
node/controlplane already cordoned
WARNING: ignoring DaemonSet-managed Pods: kube-system/kube-flannel-ds-9q9h4, kube-system/kube-proxy-vfd7
p
evicting pod default/blue-746c87566d-r8h27
evicting pod kube-system/coredns-f9fd979d6-jzt89
evicting pod default/blue-746c87566d-2c8rs
evicting pod kube-system/coredns-f9fd979d6-w8s4v
pod/blue-746c87566d-2c8rs evicted
pod/blue-746c87566d-r8h27 evicted
pod/coredns-f9fd979d6-jzt89 evicted
pod/coredns-f9fd979d6-w8s4v evicted
node/controlplane evicted
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get nodes
NAME           STATUS                     ROLES    AGE    VERSION
controlplane   Ready,SchedulingDisabled   master   153m   v1.19.0
node01         Ready                      <none>   152m   v1.19.0
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get pods -
```

01 02 03 04 05 06 07 08 09 10 11 12 13

Upgrade the `controlplane` components to exact version `v1.20.0`

Upgrade kubeadm tool (if not already), then the master components, and finally the kubelet. Practice referring to the kubernetes documentation page. Note: While upgrading kubelet, if you hit dependency issue while running the `apt-get upgrade kubelet` command, use the `apt install kubelet=1.20.0-00` command instead

Check

- controlplane Upgraded to v1.20.0
- controlplane Kubelet Upgraded to v1.20.0

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# apt update
Get:2 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:3 https://download.docker.com/linux/ubuntu bionic InRelease [64.4 kB]
Get:1 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [9383 B]
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:6 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [27.9 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:8 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packages [54.7 kB]
Get:9 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [29.8 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [3134 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [918 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [2268 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-backports/main amd64 Packages [12.2 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [12.9 kB]
Get:19 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [21.1 kB]
Get:20 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [2695 kB]
Get:21 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [1490 kB]
Get:22 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [884 kB]
Fetched 25.0 MB in 4s (6029 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
69 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@controlplane:~# apt-cache madison kubeadm
```

01 02 03 04 05 06 07 08 09 10 11 12 13

Upgrade the `controlplane` components to exact version `v1.20.0`

Upgrade kubeadm tool (if not already), then the master components, and finally the kubelet. Practice referring to the kubernetes documentation page. Note: While upgrading kubelet, if you hit dependency issue while running the `apt-get upgrade kubelet` command, use the `apt install kubelet=1.20.0-00` command instead

Check

- controlplane Upgraded to v1.20.0
- controlplane Kubelet Upgraded to v1.20.0

```
 S.L: 'deb' Debian binary tree
 S.L: 'deb-src' Debian source tree
 Idx: Debian Source Index
 Idx: Debian Package Index
 Idx: Debian Translation Index
 Idx: Debian dpkg status file
 Idx: Debian deb file
 Idx: Debian dsc file
 Idx: Debian control file
 Idx: EDSP scenario file
 Idx: EIPP scenario file
root@controlplane:~# apt-get update && \
> apt-get install -y --allow-change-held-packages kubeadm=1.20.0-00
Get:1 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [9383 B]
Hit:2 https://download.docker.com/linux/ubuntu bionic InRelease
Hit:3 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:5 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:6 http://archive.ubuntu.com/ubuntu bionic-backports InRelease
Fetched 9383 B in 1s (10.1 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following held packages will be changed:
  kubeadm
The following packages will be upgraded:
  kubeadm
1 upgraded, 0 newly installed, 0 to remove and 68 not upgraded.
Need to get 7707 kB of archives.
After this operation, 111 kB of additional disk space will be used.
Get:1 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubeadm amd64 1.20.0-00 [7707 k
B]
Fetched 7707 kB in 0s (22.8 MB/s)
debconf: delaying package configuration, since apt-utils is not installed
(Reading database ... 15149 files and directories currently installed.)
Preparing to unpack .../kubeadm_1.20.0-00_amd64.deb ...
Unpacking kubeadm (1.20.0-00) over (1.19.0-00) ...
```

01 02 03 04 05 06 07 08 09 10 11 12 13

Mark the `controlplane` node as "Schedulable" again

... Next

```
The following packages will be upgraded:
  kubectl kubelet
2 upgraded, 0 newly installed, 0 to remove and 67 not upgraded.
Need to get 26.8 MB of archives.
After this operation, 1227 kB of additional disk space will be used.
Get:1 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubectl amd64 1.20.0-00 [7942 k
B]
Get:2 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubelet amd64 1.20.0-00 [18.8 M
B]
Fetched 26.8 MB in 1s (49.7 MB/s)
debconf: delaying package configuration, since apt-utils is not installed
(Reading database ... 15149 files and directories currently installed.)
Preparing to unpack .../kubectl_1.20.0-00_amd64.deb ...
Unpacking kubectl (1.20.0-00) over (1.19.0-00) ...
Preparing to unpack .../kubelet_1.20.0-00_amd64.deb ...
/usr/sbin/policy-rc.d returned 101, not running 'stop kubelet.service'
Unpacking kubelet (1.20.0-00) over (1.19.0-00) ...
Setting up kubelet (1.20.0-00) ...
/usr/sbin/policy-rc.d returned 101, not running 'start kubelet.service'
Setting up kubectl (1.20.0-00) ...
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# sudo systemctl daemon-reload
root@controlplane:~# sudo systemctl restart kubelet
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get nodes
NAME           STATUS                     ROLES                  AGE    VERSION
controlplane   Ready,SchedulingDisabled   control-plane,master   161m   v1.20.0
node01         Ready                      <none>                 160m   v1.19.0
root@controlplane:~# k uncordon controlplane
node/controlplane uncordoned
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get nodes
NAME           STATUS   ROLES                  AGE    VERSION
controlplane   Ready    control-plane,master   161m   v1.20.0
node01         Ready    <none>                 161m   v1.19.0
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12 13

Next is the worker node. `Drain` the worker node of the workloads and mark it `UnSchedulable`

Check

- Worker node: Unschedulable

```
NAME           STATUS   ROLES                  AGE    VERSION
controlplane   Ready    control-plane,master   162m   v1.20.0
node01         Ready    <none>                 161m   v1.19.0
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k drain node01
node/node01 cordoned
error: unable to drain node "node01", aborting command...

There are pending nodes to be drained:
 node01
error: cannot delete DaemonSet-managed Pods (use --ignore-daemonsets to ignore): kube-system/kube-flanne
l-ds-k7nwt, kube-system/kube-proxy-lhvrg
root@controlplane:~# k drain node01 --ignore-daemonsets
node/node01 already cordoned
WARNING: ignoring DaemonSet-managed Pods: kube-system/kube-flannel-ds-k7nwt, kube-system/kube-proxy-lhvr
g
evicting pod kube-system/coredns-74ff55c5b-lb7p6
evicting pod default/blue-746c87566d-742dj
evicting pod default/blue-746c87566d-64ctq
evicting pod default/blue-746c87566d-ptxxp
evicting pod default/blue-746c87566d-4x2bh
evicting pod default/blue-746c87566d-mvjls
evicting pod kube-system/coredns-74ff55c5b-fqrbd
I0417 20:04:27.962015   12657 request.go:655] Throttling request took 1.09692193s, request: GET:https://
controlplane:6443/api/v1/namespaces/default/pods/blue-746c87566d-64ctq
pod/blue-746c87566d-742dj evicted
I0417 20:04:38.891517   12657 request.go:655] Throttling request took 1.027745316s, request: GET:https:/
/controlplane:6443/api/v1/namespaces/kube-system/pods/coredns-74ff55c5b-fqrbd
pod/blue-746c87566d-4x2bh evicted
pod/blue-746c87566d-ptxxp evicted
pod/blue-746c87566d-mvjls evicted
pod/blue-746c87566d-64ctq evicted
pod/coredns-74ff55c5b-lb7p6 evicted
pod/coredns-74ff55c5b-fqrbd evicted
node/node01 evicted
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# clear
```

01 02 03 04 05 06 07 08 09 10 11 12 13

Upgrade the worker node to the exact version `v1.20.0`

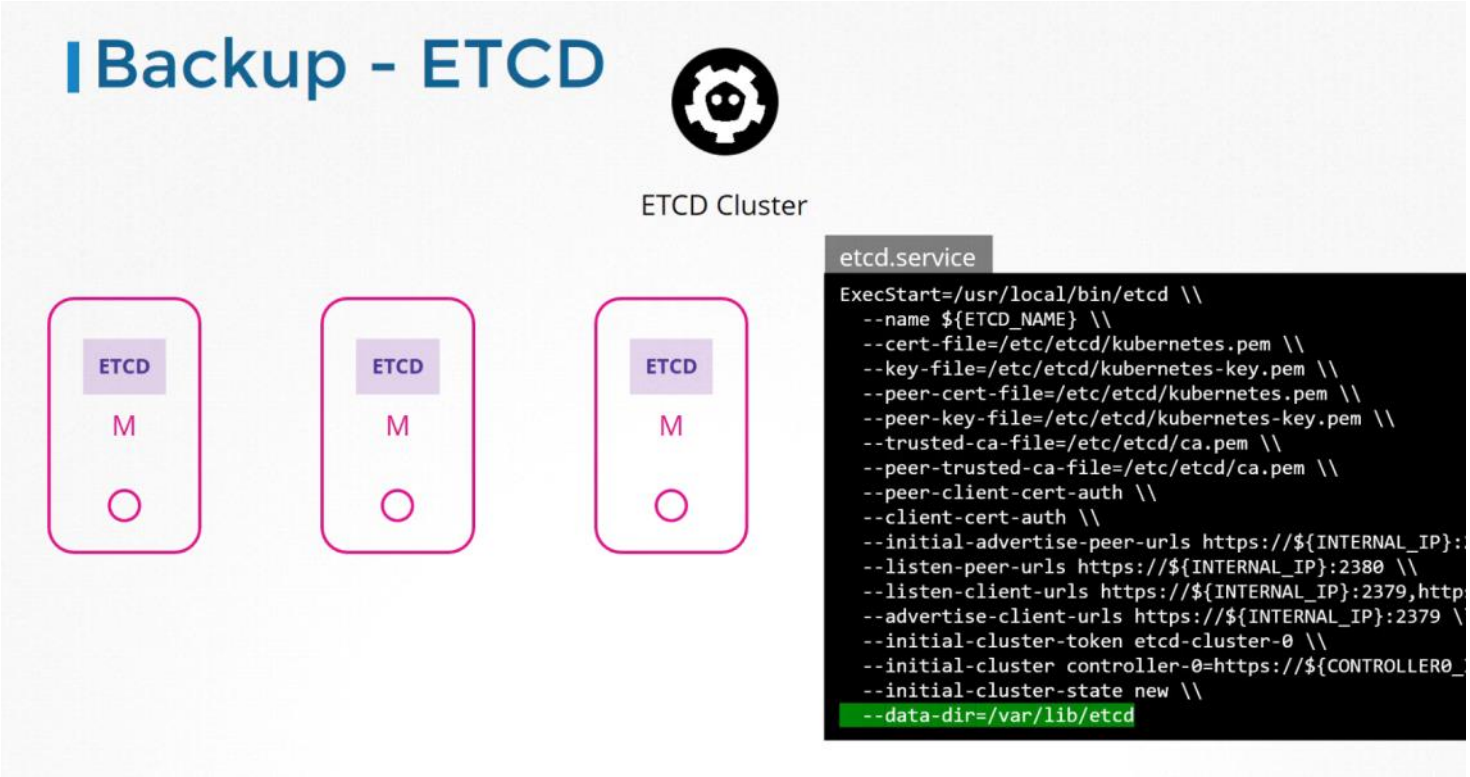
Check

- Worker Node Upgraded to v1.20.0
- Worker Node Ready

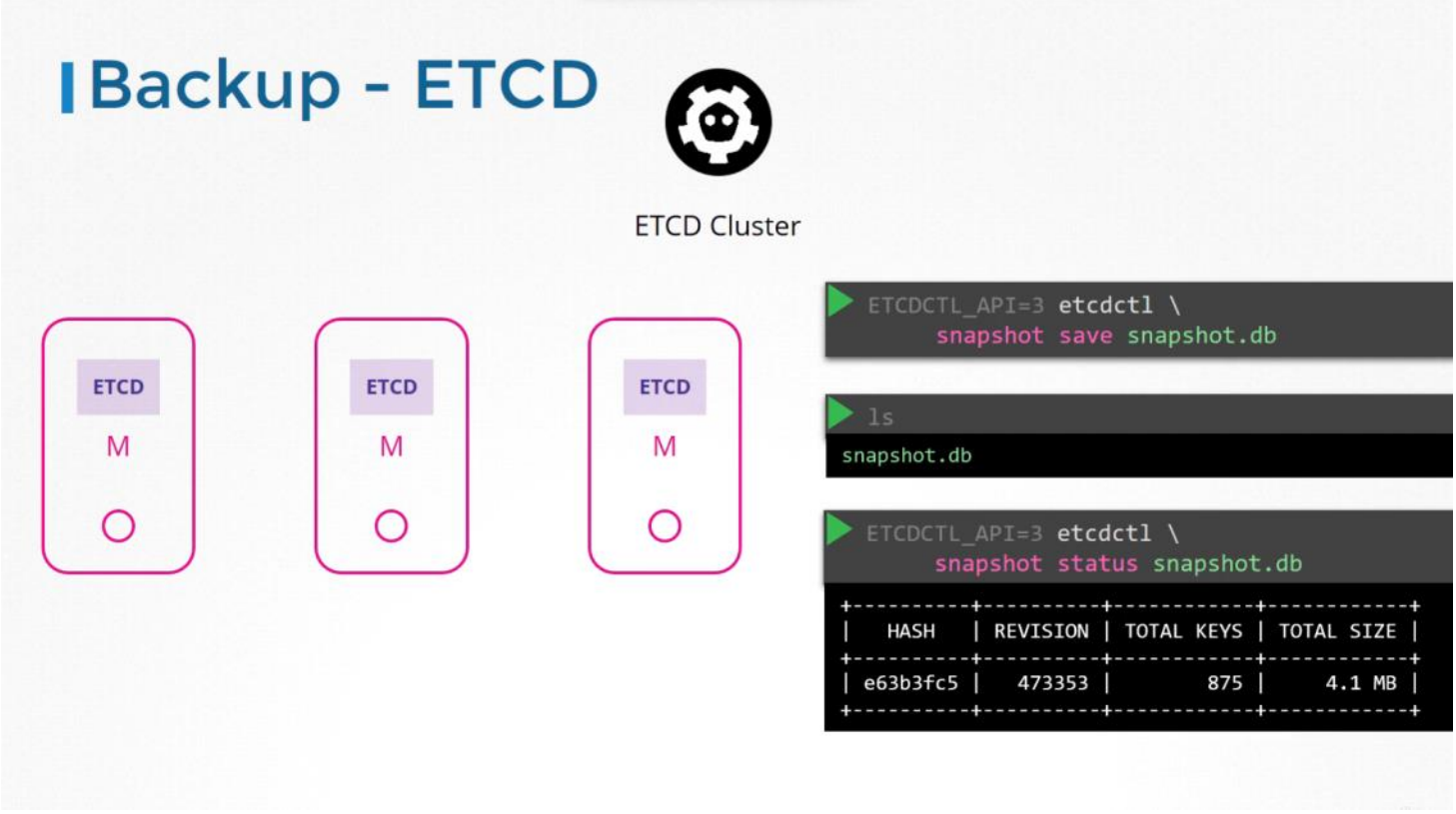
```
root@controlplane:~# k get node
NAME           STATUS                     ROLES                  AGE    VERSION
controlplane   Ready                      control-plane,master   162m   v1.20.0
node01         Ready,SchedulingDisabled   <none>                 162m   v1.19.0
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP            NODE           NOMINATED NODE
 READINESS GATES
blue-746c87566d-5djj7    1/1     Running   0          28s   10.244.0.9    controlplane   <none>
 <none>
blue-746c87566d-7gv6w    1/1     Running   0          28s   10.244.0.11   controlplane   <none>
 <none>
blue-746c87566d-j4dhw    1/1     Running   0          29s   10.244.0.6    controlplane   <none>
 <none>
blue-746c87566d-pf6vr    1/1     Running   0          28s   10.244.0.8    controlplane   <none>
 <none>
blue-746c87566d-x9dfv    1/1     Running   0          28s   10.244.0.7    controlplane   <none>
 <none>
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get nodes -o wide
NAME           STATUS                     ROLES                  AGE    VERSION   INTERNAL-IP    EXTERNA
L-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
controlplane   Ready                      control-plane,master   163m   v1.20.0   10.38.209.9    <none>
       Ubuntu 18.04.5 LTS   5.4.0-1069-gcp   docker://19.3.0
node01         Ready,SchedulingDisabled   <none>                 163m   v1.19.0   10.38.209.12   <none>
       Ubuntu 18.04.5 LTS   5.4.0-1069-gcp   docker://19.3.0
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# ssh 10.38.209.12
The authenticity of host '10.38.209.12 (10.38.209.12)' can't be established.
ECDSA key fingerprint is SHA256:QYdgRfRLyd2TpRRTdVCgvYj7ijmYgPEW05eIuimyKtE.
Are you sure you want to continue connecting (yes/no)?
```

backup and restore methods

Friday, June 28, 2024 12:12 PM



You can take backup of etcd cluster instead of taking backup of all resources.



Restore - ETCD

ETCD Cluster

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

```
ls
snapshot.db
```

```
service kube-apiserver stop
Service kube-apiserver stopped
```

```
ETCDCTL_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
```

Restore - ETCD

ETCD Cluster

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

```
ls
snapshot.db
```

```
service kube-apiserver stop
Service kube-apiserver stopped
```

```
ETCDCTL_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
```

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

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

Working with ETCDCTL

etcdctl is a command line client for **etcd**.

In all our Kubernetes Hands-on labs, the ETCD key-value database is deployed as a static pod on the master. The version used is v3.
To make use of etcdctl for tasks such as back up and restore, make sure that you set the ETCDCTL_API to 3.

You can do this by exporting the variable ETCDCTL_API prior to using the etcdctl client. This can be done as follows:
export ETCDCTL_API=3
On the **Master Node**:

```
master $ export ETCDCTL_API=3
master $ etcdctl version
etcdctl version: 3.3.13
API version: 3.3
master $
```

To see all the options for a specific sub-command, make use of the **-h or --help** flag.

For example, if you want to take a snapshot of etcd, use:
etcdctl snapshot save -h and keep a note of the mandatory global options.

Since our ETCD database is TLS-Enabled, the following options are mandatory:

--cacert	verify certificates of TLS-enabled secure servers using this CA bundle
--cert	identify secure client using this TLS certificate file
--endpoints=[127.0.0.1:2379]	This is the default as ETCD is running on master node and exposed on localhost 2379.
--key	identify secure client using this TLS key file

Similarly use the help option for **snapshot restore** to see all available options for restoring the backup.
etcdctl snapshot restore -h
For a detailed explanation on how to make use of the etcdctl command line tool and work with the -h flags, check out the solution video for the Backup and Restore Lab.

Practice- Backup and Restore

Friday, June 28, 2024 1:48 PM

1 2 3 4 5 6 7 8 9

We have a working kubernetes cluster with a set of applications running. Let us first explore the setup. How many deployments exist in the cluster?

```
root@controlplane ~ ➜  k get deploy
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
blue   3/3     3            3           18s
red    2/2     2            2           18s

root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9

What is the version of ETCD running on the cluster? Check the ETCD Pod or Process

"3.4.13"

"2.1.0"

"2.3.4"

"3.5.1"

```
root@controlplane ~ ➜  k get deploy
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
blue   3/3     3            3           18s
red    2/2     2            2           18s

root@controlplane ~ ➜  k get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
coredns-64897985d-btbqw                    1/1     Running   0          20m
coredns-64897985d-c5r9t                    1/1     Running   0          20m
etcd-controlplane                          1/1     Running   0          21m
kube-apiserver-controlplane                1/1     Running   0          21m
kube-controller-manager-controlplane       1/1     Running   0          21m
kube-flannel-ds-2bt8m                      1/1     Running   0          20m
kube-proxy-6bcbb                           1/1     Running   0          20m
kube-scheduler-controlplane                1/1     Running   0          21m

root@controlplane ~ ➜  k describe pod etcd-controlplane -n kube-system
```

```
Status:               Running
IP:                   10.44.214.9
IPs:
  IP:           10.44.214.9
Controlled By:  Node/controlplane
Containers:
  etcd:
    Container ID:  docker://5930818c18aafc4b00d3eb301cec4427e88249a2ab5291743f05bfa4f5dbf4b7
    Image:         k8s.gcr.io/etcd:3.5.1-0
    Image ID:      docker-pullable://k8s.gcr.io/etcd@sha256:64b9ea357325d5db9f8a723dcf503b5a449177b17ac8
7d69481e126bb724c263
    Port:          <none>
    Host Port:     <none>
    Command:
      etcd
      --advertise-client-urls=https://10.44.214.9:2379
      --cert-file=/etc/kubernetes/pki/etcd/server.crt
      --client-cert-auth=true
      --data-dir=/var/lib/etcd
      --initial-advertise-peer-urls=https://10.44.214.9:2380
      --initial-cluster=controlplane=https://10.44.214.9:2380
      --key-file=/etc/kubernetes/pki/etcd/server.key
      --listen-client-urls=https://127.0.0.1:2379,https://10.44.214.9:2379
```

Look at the ETCD Logs using the command kubectl logs etcd-controlplane -n kube-system or check the image used by the ETCD pod: kubectl describe pod etcd-controlplane -n kube-system	

1 2 3 4 5 6 7 8 9

At what address can you reach the ETCD cluster from the controlplane node?
Check the ETCD Service configuration in the ETCD POD

https://127.0.0.1:2379

https://127.0.0.1:2380

https://10.96.0.1:8080

https://10.96.0.1:2379

```
  etcd:
    Container ID:  docker://5930818c18aafc4b00d3eb301cec4427e88249a2ab5291743f05bfa4f5dbf4b7
    Image:         k8s.gcr.io/etcd:3.5.1-0
    Image ID:      docker-pullable://k8s.gcr.io/etcd@sha256:64b9ea357325d5db9f8a723dcf503b5a449177b17ac8
7d69481e126bb724c263
    Port:          <none>
    Host Port:     <none>
    Command:
      etcd
      --advertise-client-urls=https://10.44.214.9:2379
      --cert-file=/etc/kubernetes/pki/etcd/server.crt
      --client-cert-auth=true
      --data-dir=/var/lib/etcd
      --initial-advertise-peer-urls=https://10.44.214.9:2380
      --initial-cluster=controlplane=https://10.44.214.9:2380
      --key-file=/etc/kubernetes/pki/etcd/server.key
      --listen-client-urls=https://127.0.0.1:2379,https://10.44.214.9:2379
      --listen-metrics-urls=http://127.0.0.1:2381
      --listen-peer-urls=https://10.44.214.9:2380
      --name=controlplane
      --peer-cert-file=/etc/kubernetes/pki/etcd/peer.crt
      --peer-client-cert-auth=true
      --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
      --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
      --snapshot-count=10000
      --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
    State:          Running
      Started:      Sun, 17 Apr 2022 20:19:42 +0000
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     100Mi
    Liveness:     http-get http://127.0.0.1:2381/health delay=10s timeout=15s period=10s #success=1 #fai
lure=8
    Startup:      http-get http://127.0.0.1:2381/health delay=10s timeout=15s period=10s #success=1 #fai
lure=24
    Environment:  <none>
    Mounts:
      /etc/kubernetes/pki/etcd from etcd-certs (rw)
```

1 2 3 4 5 6 7 8 9

Where is the ETCD server certificate file located?
Note this path down as you will need to use it later

/etc/kubernetes/pki/etcd/server.crt

/etc/kubernetes/pki/server.crt

/etc/kubernetes/pki/etcd/ca.crt

/etc/kubernetes/pki/etcd/peer.crt


```
  etcd:
    Container ID:  docker://5930818c18aafc4b00d3eb301cec4427e88249a2ab5291743f05bfa4f5dbf4b7
    Image:         k8s.gcr.io/etcd:3.5.1-0
    Image ID:      docker-pullable://k8s.gcr.io/etcd@sha256:64b9ea357325d5db9f8a723dcf503b5a449177b17ac8
7d69481e126bb724c263
    Port:          <none>
    Host Port:     <none>
    Command:
      etcd
      --advertise-client-urls=https://10.44.214.9:2379
      --cert-file=/etc/kubernetes/pki/etcd/server.crt
      --client-cert-auth=true
      --data-dir=/var/lib/etcd
      --initial-advertise-peer-urls=https://10.44.214.9:2380
      --initial-cluster=controlplane=https://10.44.214.9:2380
      --key-file=/etc/kubernetes/pki/etcd/server.key
      --listen-client-urls=https://127.0.0.1:2379,https://10.44.214.9:2379
      --listen-metrics-urls=http://127.0.0.1:2381
      --listen-peer-urls=https://10.44.214.9:2380
      --name=controlplane
      --peer-cert-file=/etc/kubernetes/pki/etcd/peer.crt
      --peer-client-cert-auth=true
      --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
      --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
      --snapshot-count=10000
      --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
    State:          Running
      Started:      Sun, 17 Apr 2022 20:19:42 +0000
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     100Mi
    Liveness:     http-get http://127.0.0.1:2381/health delay=10s timeout=15s period=10s #success=1 #fai
lure=8
    Startup:      http-get http://127.0.0.1:2381/health delay=10s timeout=15s period=10s #success=1 #fai
lure=24
    Environment:  <none>
    Mounts:
      /etc/kubernetes/pki/etcd from etcd-certs (rw)
```

1 2 3 4 5 6 7 8 9

Where is the ETCD CA Certificate file located?
Note this path down as you will need to use it later.

/etc/kubernetes/pki/etcd/peer.crt

/etc/kubernetes/pki/etcd/ca.crt

/etc/kubernetes/pki/ca.crt

/etc/kubernetes/pki/etcd/ca.key

```
  etcd:
    Container ID:  docker://5930818c18aafc4b00d3eb301cec4427e88249a2ab5291743f05bfa4f5dbf4b7
    Image:         k8s.gcr.io/etcd:3.5.1-0
    Image ID:      docker-pullable://k8s.gcr.io/etcd@sha256:64b9ea357325d5db9f8a723dcf503b5a449177b17ac8
7d69481e126bb724c263
    Port:          <none>
    Host Port:     <none>
    Command:
      etcd
      --advertise-client-urls=https://10.44.214.9:2379
      --cert-file=/etc/kubernetes/pki/etcd/server.crt
      --client-cert-auth=true
      --data-dir=/var/lib/etcd
      --initial-advertise-peer-urls=https://10.44.214.9:2380
      --initial-cluster=controlplane=https://10.44.214.9:2380
      --key-file=/etc/kubernetes/pki/etcd/server.key
      --listen-client-urls=https://127.0.0.1:2379,https://10.44.214.9:2379
      --listen-metrics-urls=http://127.0.0.1:2381
      --listen-peer-urls=https://10.44.214.9:2380
      --name=controlplane
      --peer-cert-file=/etc/kubernetes/pki/etcd/peer.crt
      --peer-client-cert-auth=true
      --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
      --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
      --snapshot-count=10000
      --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
    State:          Running
      Started:      Sun, 17 Apr 2022 20:19:42 +0000
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     100Mi
    Liveness:     http-get http://127.0.0.1:2381/health delay=10s timeout=15s period=10s #success=1 #fai
lure=8
    Startup:      http-get http://127.0.0.1:2381/health delay=10s timeout=15s period=10s #success=1 #fai
lure=24
    Environment:  <none>
    Mounts:
      /etc/kubernetes/pki/etcd from etcd-certs (rw)
```

1 2 3 4 5 6 7 8 9

The master node in our cluster are planned for a regular maintenance reboot tonight. While we do not anticipate anything to go wrong, we are required to take the necessary backups. Take a snapshot of the ETCD database using the built-in snapshot functionality.

Store the backup file at location `/opt/snapshot-pre-boot.db`

Check

- Backup ETCD to /opt/snapshot-pre-boot.db

```
        status  Gets backend snapshot status of a given file

OPTIONS:
  -h, --help[=false]    help for snapshot

GLOBAL OPTIONS:
      --cacert=""                       verify certificates of TLS-enabled secure servers using
this CA bundle
      --cert=""                         identify secure client using this TLS certificate file
      --command-timeout=5s              timeout for short running command (excluding dial timeou
t)
      --debug[=false]                   enable client-side debug logging
      --dial-timeout=2s                 dial timeout for client connections
  -d, --discovery-srv=""                domain name to query for SRV records describing cluster
endpoints
      --endpoints=[127.0.0.1:2379]      gRPC endpoints
      --hex[=false]                     print byte strings as hex encoded strings
      --insecure-discovery[=true]       accept insecure SRV records describing cluster endpoints
      --insecure-skip-tls-verify[=false]  skip server certificate verification
      --insecure-transport[=true]       disable transport security for client connections
      --keepalive-time=2s               keepalive time for client connections
      --keepalive-timeout=6s            keepalive timeout for client connections
      --key=""                          identify secure client using this TLS key file
      --user=""                         username[:password] for authentication (prompt if passwo
rd is not supplied)
  -w, --write-out="simple"              set the output format (fields, json, protobuf, simple, t
able)


root@controlplane ~ ➜  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.key \
> /opt/snapshot-pre-boot.db
Snapshot saved at /opt/snapshot-pre-boot.db

root@controlplane ~ ➜  ls /opt/snapshot-pre-boot.db
/opt/snapshot-pre-boot.db

root@controlplane ~ ➜
```

Task Solution 57:48

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

How many `clusters` are defined in the kubeconfig on the `student-node` ?

You can make use of the `kubectl config` command.

1

3

0

2

4

Terminal 1 +

```
student-node ~ ➜ kubectl get node
NAME                   STATUS   ROLES           AGE   VERSION
cluster1-controlplane  Ready    control-plane   51m   v1.24.0
cluster1-node01        Ready    <none>          50m   v1.24.0

student-node ~ ➜ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://cluster1-controlplane:6443
  name: cluster1
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://192.33.162.10:6443
  name: cluster2
contexts:
- context:
    cluster: cluster1
    user: cluster1
  name: cluster1
- context:
    cluster: cluster2
    user: cluster2
  name: cluster2
current-context: cluster1
kind: Config
preferences: {}
users:
- name: cluster1
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
- name: cluster2
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED

student-node ~ ➜
```

Task Solution 57:06

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

How many nodes (both controlplane and worker) are part of `cluster1` ?

Make sure to switch the context to `cluster1` :

```
kubectl config use-context cluster1
```

4

2

0

1

3

Terminal 1 +

```
student-node ~ ➜ kubectl get nodes
NAME                   STATUS   ROLES           AGE   VERSION
cluster1-controlplane  Ready    control-plane   52m   v1.24.0
cluster1-node01        Ready    <none>          51m   v1.24.0

student-node ~ ➜
```

Task Solution 56:26

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

What is the name of the controlplane node in `cluster2` ?

Make sure to switch the context to `cluster2` :

```
kubectl config use-context cluster2
```

cluster1-controlplane

cluster2-master

cluster2-controlplane

controlplane

cluster2controlplane

controlplane-cluster2

Terminal 1

```
student-node ~ ➜ kubectl get nodes
NAME                    STATUS   ROLES           AGE   VERSION
cluster1-controlplane   Ready    control-plane   52m   v1.24.0
cluster1-node01         Ready    <none>          51m   v1.24.0

student-node ~ ➜ kubectl config use-context cluster2
Switched to context "cluster2".

student-node ~ ➜ kubectl get nodes
NAME                    STATUS   ROLES           AGE   VERSION
cluster2-controlplane   Ready    control-plane   53m   v1.24.0
cluster2-node01         Ready    <none>          52m   v1.24.0

student-node ~ ➜
```

Task Solution 55:41

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

You can SSH to all the nodes (of both clusters) from the `student-node` .

For example:

```
student-node ~ ➜  ssh cluster1-controlplane
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1086-gcp x86_6

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
This system has been minimized by removing packages and conte
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command

cluster1-controlplane ~ ➜
```

To get back to the `student` node, use the `logout` or `exit` command, or, hit `Control +D`

```
cluster1-controlplane ~ ➜  logout
```

Terminal 1

```
student-node ~ ➜ kubectl get nodes
NAME                    STATUS   ROLES           AGE   VERSION
cluster1-controlplane   Ready    control-plane   52m   v1.24.0
cluster1-node01         Ready    <none>          51m   v1.24.0

student-node ~ ➜ kubectl config use-context cluster2
Switched to context "cluster2".

student-node ~ ➜ kubectl get nodes
NAME                    STATUS   ROLES           AGE   VERSION
cluster2-controlplane   Ready    control-plane   53m   v1.24.0
cluster2-node01         Ready    <none>          52m   v1.24.0

student-node ~ ➜ ssh cluster2-controlplane
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1103-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

cluster2-controlplane ~ ➜ exit
logout
Connection to cluster2-controlplane closed.

student-node ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

How is `ETCD` configured for `cluster1` ?

Remember, you can access the clusters from `student-node` using the `kubectl` tool.

You can also `ssh` to the cluster nodes from the `student-node` .

Make sure to switch the context to `cluster1` :

```
kubectl config use-context cluster1
```

No ETCD

External ETCD

Stacked ETCD

```
student-node ~ ➜ kubectl get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS      AGE
coredns-6d4b75cb6d-pmrrv                   1/1     Running   0             55m
coredns-6d4b75cb6d-v6mvc                   1/1     Running   0             55m
etcd-cluster1-controlplane                 1/1     Running   0             55m
kube-apiserver-cluster1-controlplane       1/1     Running   0             55m
kube-controller-manager-cluster1-controlplane   1/1     Running   0             55m
kube-proxy-4zw7h                           1/1     Running   0             54m
kube-proxy-zhb52                           1/1     Running   0             55m
kube-scheduler-cluster1-controlplane       1/1     Running   0             55m
weave-net-2zg84                            2/2     Running   0             54m
weave-net-hsx4z                            2/2     Running   1 (55m ago)   55m

student-node ~ ➜ kubectl describe pod kube-apiserver-cluster1-controlplane -n kube-system
```

Task Solution 53:01

Terminal 1 +

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

How is `ETCD` configured for `cluster1` ?

Remember, you can access the clusters from `student-node` using the `kubectl` tool.

You can also `ssh` to the cluster nodes from the `student-node` .

Make sure to switch the context to `cluster1` :

```
kubectl config use-context cluster1
```

No ETCD

External ETCD

```
8c955
    Port:          <none>
    Host Port:     <none>
    Command:
      kube-apiserver
      --advertise-address=192.33.162.8
      --allow-privileged=true
      --authorization-mode=Node,RBAC
      --client-ca-file=/etc/kubernetes/pki/ca.crt
      --enable-admission-plugins=NodeRestriction
      --enable-bootstrap-token-auth=true
      --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
      --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
      --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
      --etcd-servers=https://127.0.0.1:2379
      --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
      --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
      --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
      --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
      --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
      --requestheader-allowed-names=front-proxy-client
      --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
      --requestheader-extra-headers-prefix=X-Remote-Extra-
      --requestheader-group-headers=X-Remote-Group
      --requestheader-username-headers=X-Remote-User
      --secure-port=6443
      --service-account-issuer=https://kubernetes.default.svc.cluster.local
      --service-account-key-file=/etc/kubernetes/pki/sa.pub
      --service-account-signing-key-file=/etc/kubernetes/pki/sa.key
      --service-cluster-ip-range=10.96.0.0/12
      --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
      --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
    State:          Running
      Started:      Thu, 27 Apr 2023 14:07:24 +0000
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        250m
    Liveness:     http-get https://192.33.162.8:6443/livez delay=10s timeout=15s period=10s #success=1 #failure
=8
    Readiness:    http-get https://192.33.162.8:6443/readyz delay=0s timeout=15s period=1s #success=1 #fail@pe=
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

How is `ETCD` configured for `cluster2` ?

Remember, you can access the clusters from `student-node` using the `kubectl` tool.

You can also `ssh` to the cluster nodes from the `student-node` .

Make sure to switch the context to `cluster2` :

```
kubectl config use-context cluster2
```

No ETCD

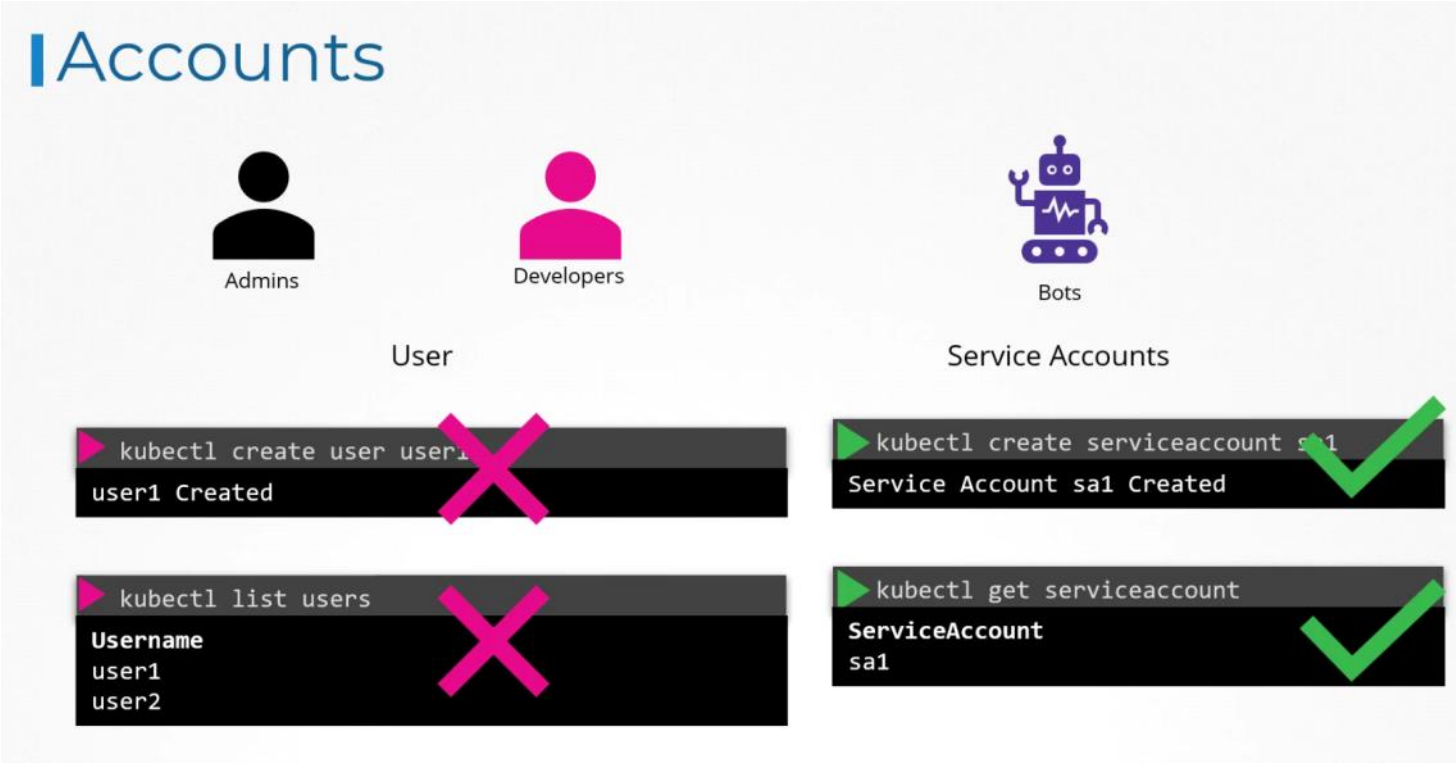
Stacked ETCD

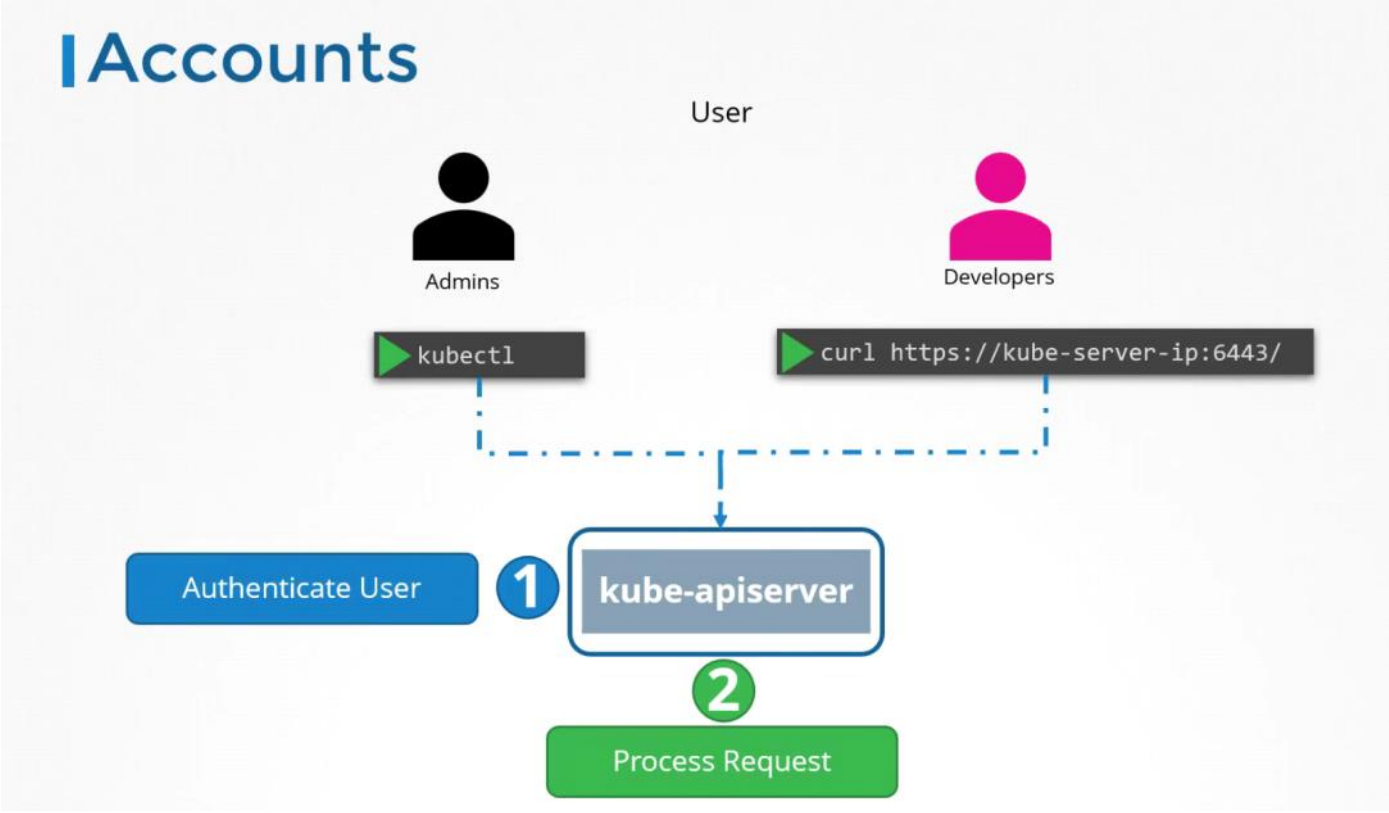
External ETCD

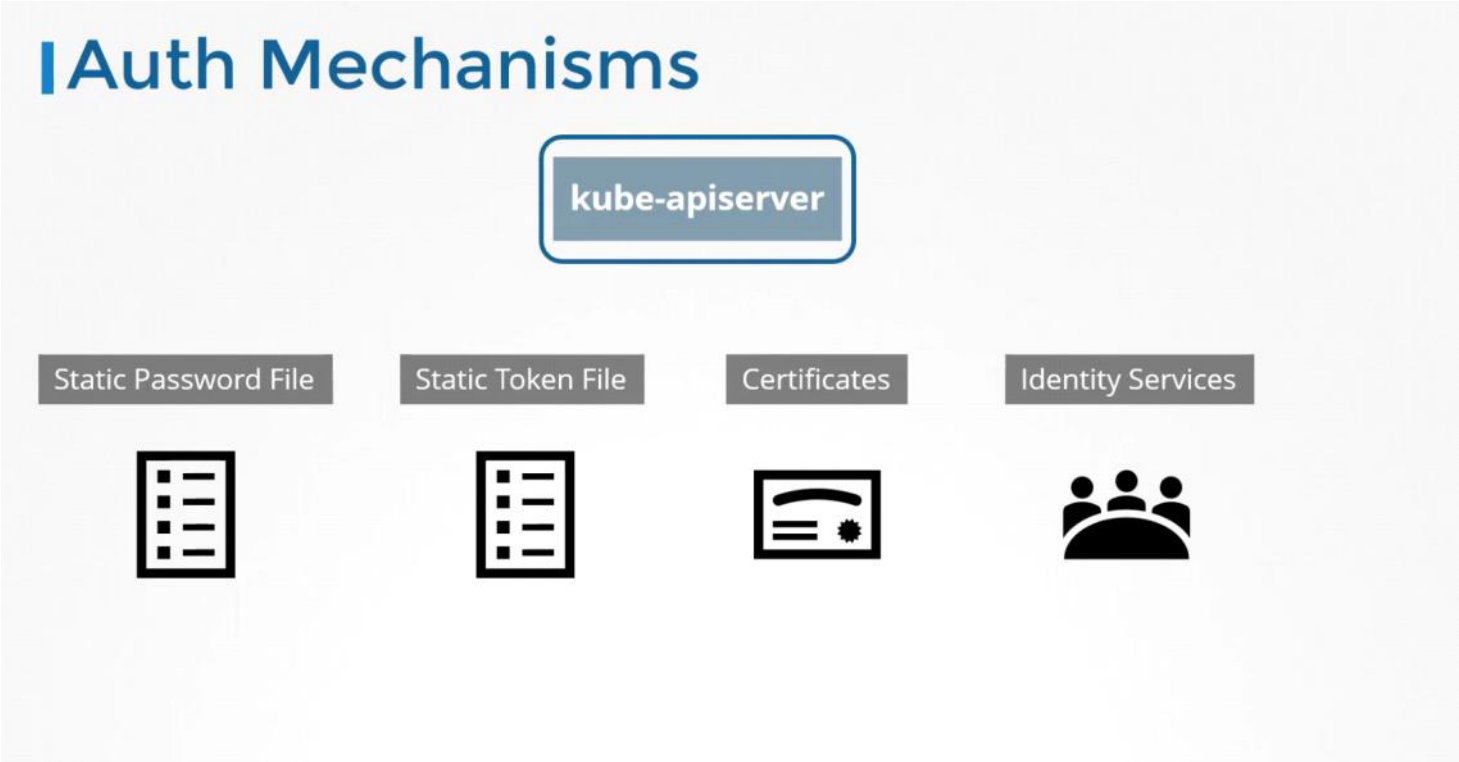
```
    Image:          k8s.gcr.io/kube-apiserver:v1.24.0
    Image ID:       k8s.gcr.io/kube-apiserver@sha256:a04522b882e919de6141b47d72393fb01226c78e7388400f96619822255
8c955
    Port:           <none>
    Host Port:      <none>
    Command:
      kube-apiserver
      --advertise-address=192.33.162.10
      --allow-privileged=true
      --authorization-mode=Node,RBAC
      --client-ca-file=/etc/kubernetes/pki/ca.crt
      --enable-admission-plugins=NodeRestriction
      --enable-bootstrap-token-auth=true
      --etcd-cafile=/etc/kubernetes/pki/etcd/ca.pem
      --etcd-certfile=/etc/kubernetes/pki/etcd/etcd.pem
      --etcd-keyfile=/etc/kubernetes/pki/etcd/etcd-key.pem
      --etcd-servers=https://192.33.162.21:2379
      --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
      --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
      --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
      --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
      --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
      --requestheader-allowed-names=front-proxy-client
      --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
      --requestheader-extra-headers-prefix=X-Remote-Extra-
      --requestheader-group-headers=X-Remote-Group
      --requestheader-username-headers=X-Remote-User
      --secure-port=6443
      --service-account-issuer=https://kubernetes.default.svc.cluster.local
      --service-account-key-file=/etc/kubernetes/pki/sa.pub
      --service-account-signing-key-file=/etc/kubernetes/pki/sa.key
      --service-cluster-ip-range=10.96.0.0/12
      --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
      --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
    State:          Running
      Started:      Thu, 27 Apr 2023 14:06:58 +0000
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        250m
    Liveness:     http-get https://192.33.162.10:6443/livez delay=10s timeout=15s period=10s #success=1 #failur
e=8
    Readiness:    http-get https://192.33.162.10:6443/readyz delay=0s timeout=15s period=1s #success=1 #failure
```

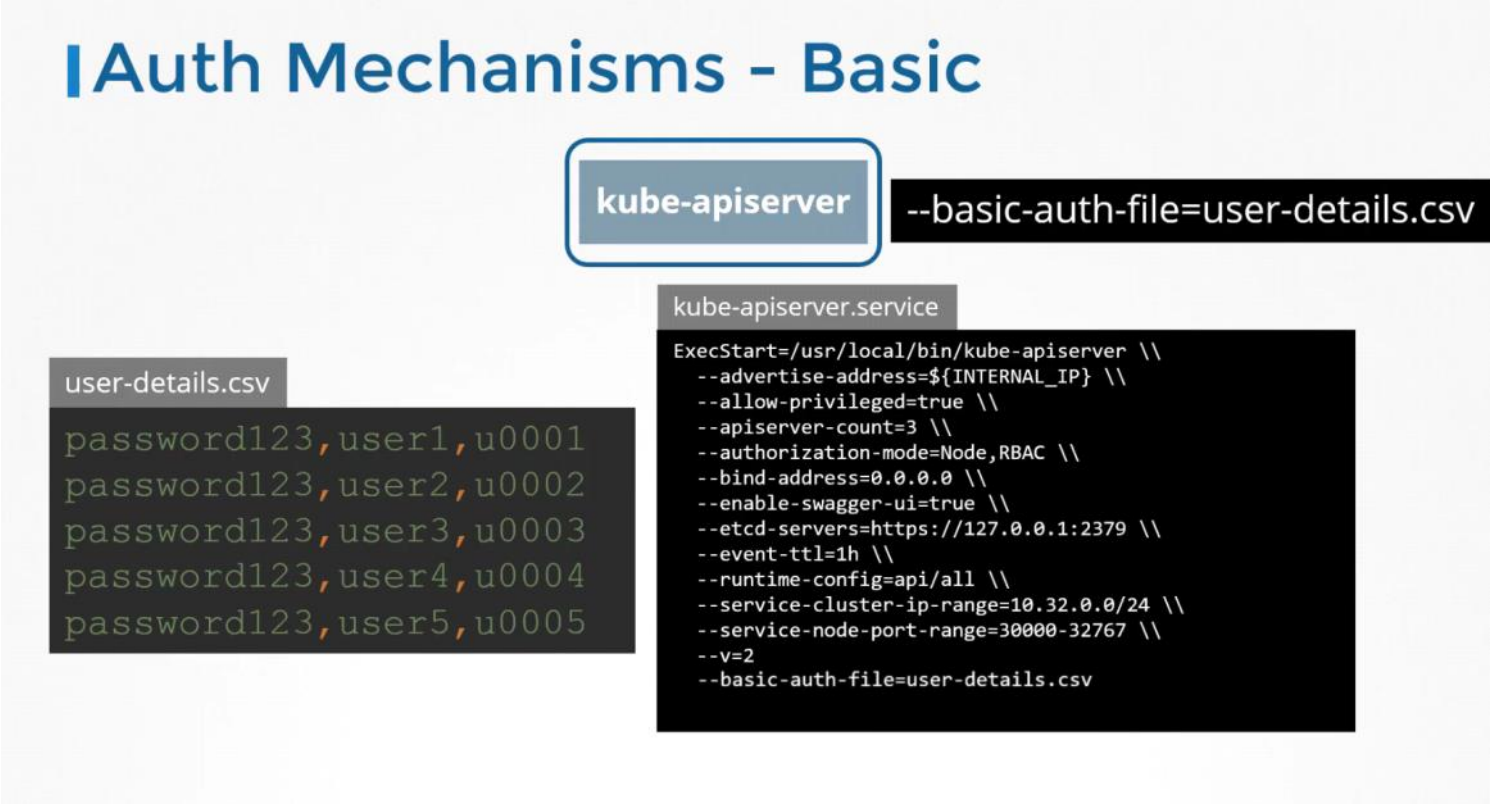
Authentication

Friday, June 28, 2024 3:26 PM


Accounts
Admins
Developers
Bots
User
Service Accounts
kubectl create user user1
user1 Created
kubectl create serviceaccount sa1
Service Account sa1 Created
kubectl list users
Username
user1
user2
kubectl get serviceaccount
ServiceAccount
sa1


Accounts
User
Admins
Developers
kubectl
curl https://kube-server-ip:6443/
Authenticate User
1
kube-apiserver
2
Process Request





static password and token file.

Authenticate User

```
curl -v -k https://master-node-ip:6443/api/v1/pods -u "user1:password123"
```


```
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/pods",
    "resourceVersion": "3594"
  },
 "items": [
    {
      "metadata": {
        "name": "nginx-64f497f8fd-krkg6",
        "generateName": "nginx-64f497f8fd-",
        "namespace": "default",
        "selfLink": "/api/v1/namespaces/default/pods/nginx-64f497f8fd-krkg6",
        "uid": "77dd7dfb-2914-11e9-b468-0242ac11006b",
        "resourceVersion": "3569",
        "creationTimestamp": "2019-02-05T07:05:49Z",
        "labels": {
          "pod-template-hash": "2090539498",
          "run": "nginx"
```


Auth Mechanisms - Basic

Static Password File

user-details.csv

```
password123,user1,u0001,group1
password123,user2,u0002,group1
password123,user3,u0003,group2
password123,user4,u0004,group2
password123,user5,u0005,group2
```

Static Token File

user-token-details.csv

```
KpjCVbI7rCFAHYPkByTIzRb7gu1cUc4B,user10,u0010,group1
rJjncHmvtXHc6MlWQddhtvNyyhgTdxSC,user11,u0011,group1
mjpOFIEiFOkL9toikaRNtt59ePtczZSq,user12,u0012,group2
PG41IXhs7QjqwWkmBkvgGT9glOyUqZij,user13,u0013,group2
```

```
--token-auth-file=user-token-details.csv
```

```
curl -v -k https://master-node-ip:6443/api/v1/pods --header "Authorization: Bearer KpjCVbI7
```

Note

- This is not a recommended authentication mechanism
- Consider volume mount while providing the auth file in a kubeadm setup
- Setup Role Based Authorization for the new users

basic authentication

Thursday, August 22, 2024 11:00 PM

Article on Setting up Basic Authentication

Setup basic authentication on Kubernetes (Deprecated in 1.19)

> Note: This is not recommended in a production environment. This is only for learning purposes. Also note that this approach is deprecated in Kubernetes version 1.19 and is no longer available in later releases

Follow the below instructions to configure basic authentication in a kubeadm setup.

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

```
# User File Contents
password123,user1,u0001
password123,user2,u0002
password123,user3,u0003
password123,user4,u0004
password123,user5,u0005
```

Edit the kube-apiserver static pod configured by kubeadm to pass in the user details.

The file is located at /etc/kubernetes/manifests/kube-apiserver.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
      <content-hidden>
    image: k8s.gcr.io/kube-apiserver-amd64:v1.11.3
    name: kube-apiserver
    volumeMounts:
    - mountPath: /tmp/users
      name: usr-details
      readOnly: true
  volumes:
  - hostPath:
      path: /tmp/users
      type: DirectoryOrCreate
    name: usr-details
```

Modify the kube-apiserver startup options to include the basic-auth file

```
apiVersion: v1
```

```
2. kind: Pod
3. metadata:
4.   creationTimestamp: null
5.   name: kube-apiserver
6.   namespace: kube-system
7. spec:
8.   containers:
9.   - command:
10.     - kube-apiserver
11.     - --authorization-mode=Node,RBAC
12.       <content-hidden>
13.     - --basic-auth-file=/tmp/users/user-details.csv
```

Create the necessary roles and role bindings for these users:

```
1. ---
2. kind: Role
3. apiVersion: rbac.authorization.k8s.io/v1
4. metadata:
5.   namespace: default
6.   name: pod-reader
7. rules:
8. - apiGroups: [""] # "" indicates the core API group
9.   resources: ["pods"]
10.   verbs: ["get", "watch", "list"]
11.
12. ---
13. # This role binding allows "jane" to read pods in the "default" namespace.
14. kind: RoleBinding
15. apiVersion: rbac.authorization.k8s.io/v1
16. metadata:
17.   name: read-pods
18.   namespace: default
19. subjects:
20. - kind: User
21.   name: user1 # Name is case sensitive
22.   apiGroup: rbac.authorization.k8s.io
23. roleRef:
24.   kind: Role #this must be Role or ClusterRole
25.   name: pod-reader # this must match the name of the Role or ClusterRole you wish to
    bind to
26.   apiGroup: rbac.authorization.k8s.io
```

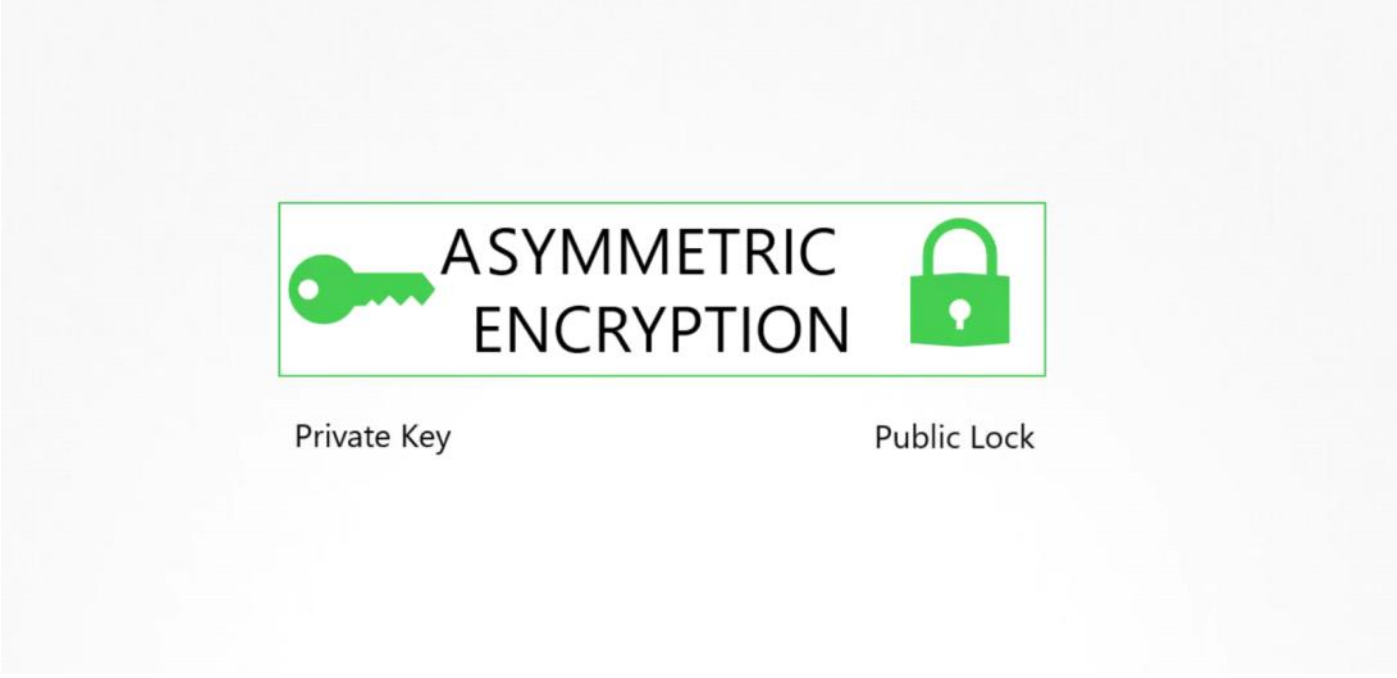
Once created, you may authenticate into the kube-api server using the users credentials

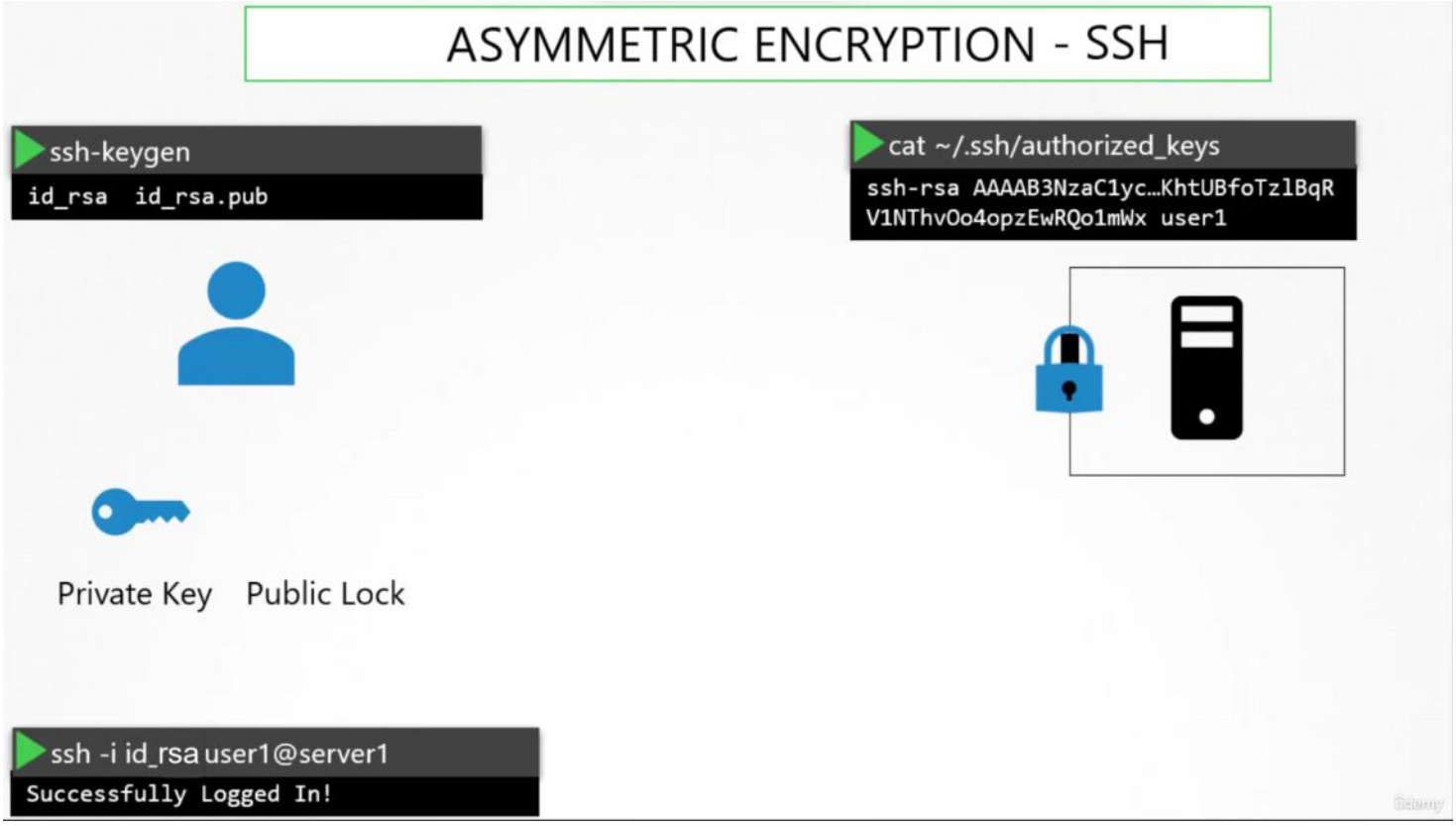
curl -v -k https://localhost:6443/api/v1/pods -u "user1:password123"

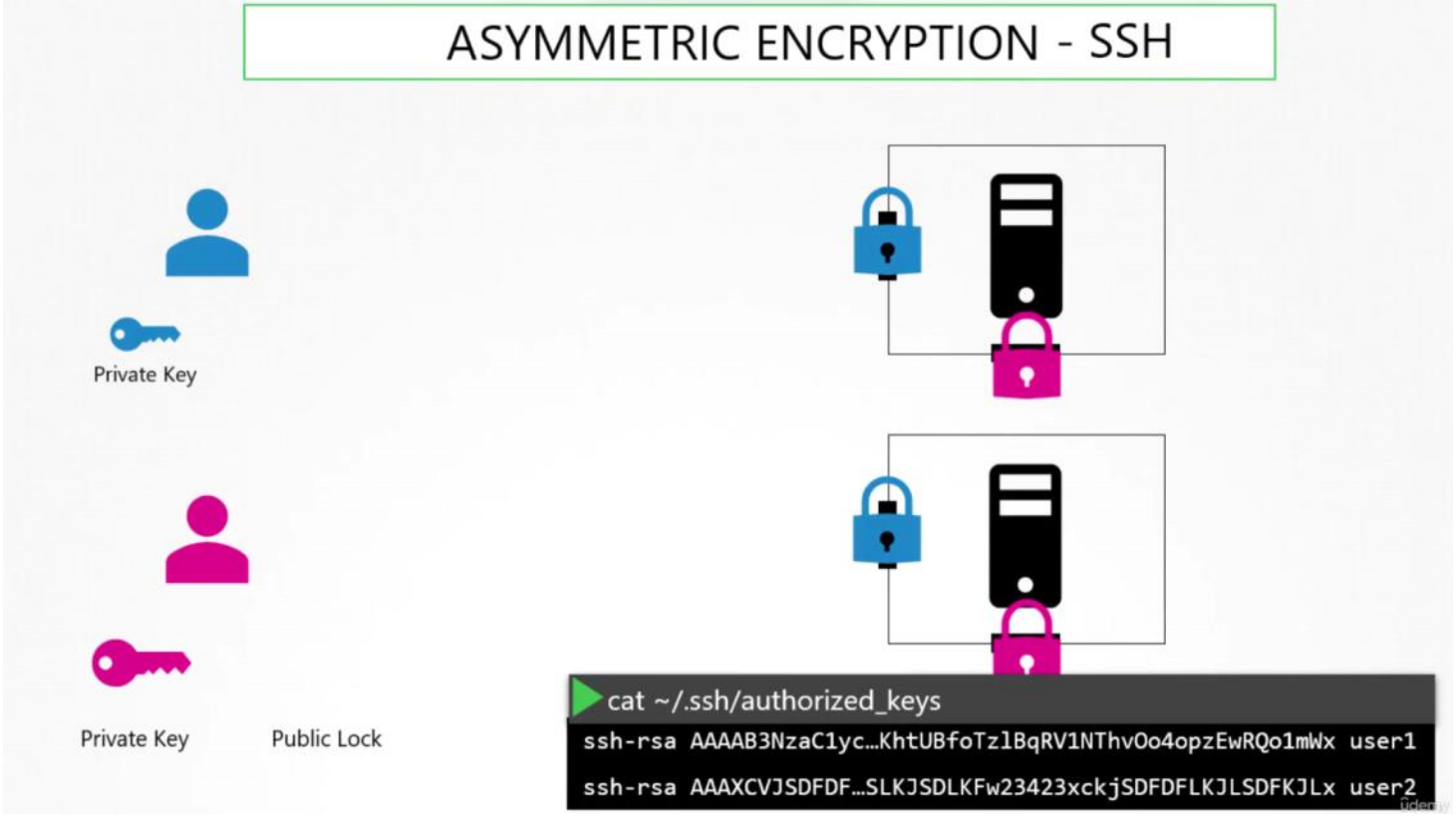
TLS basics

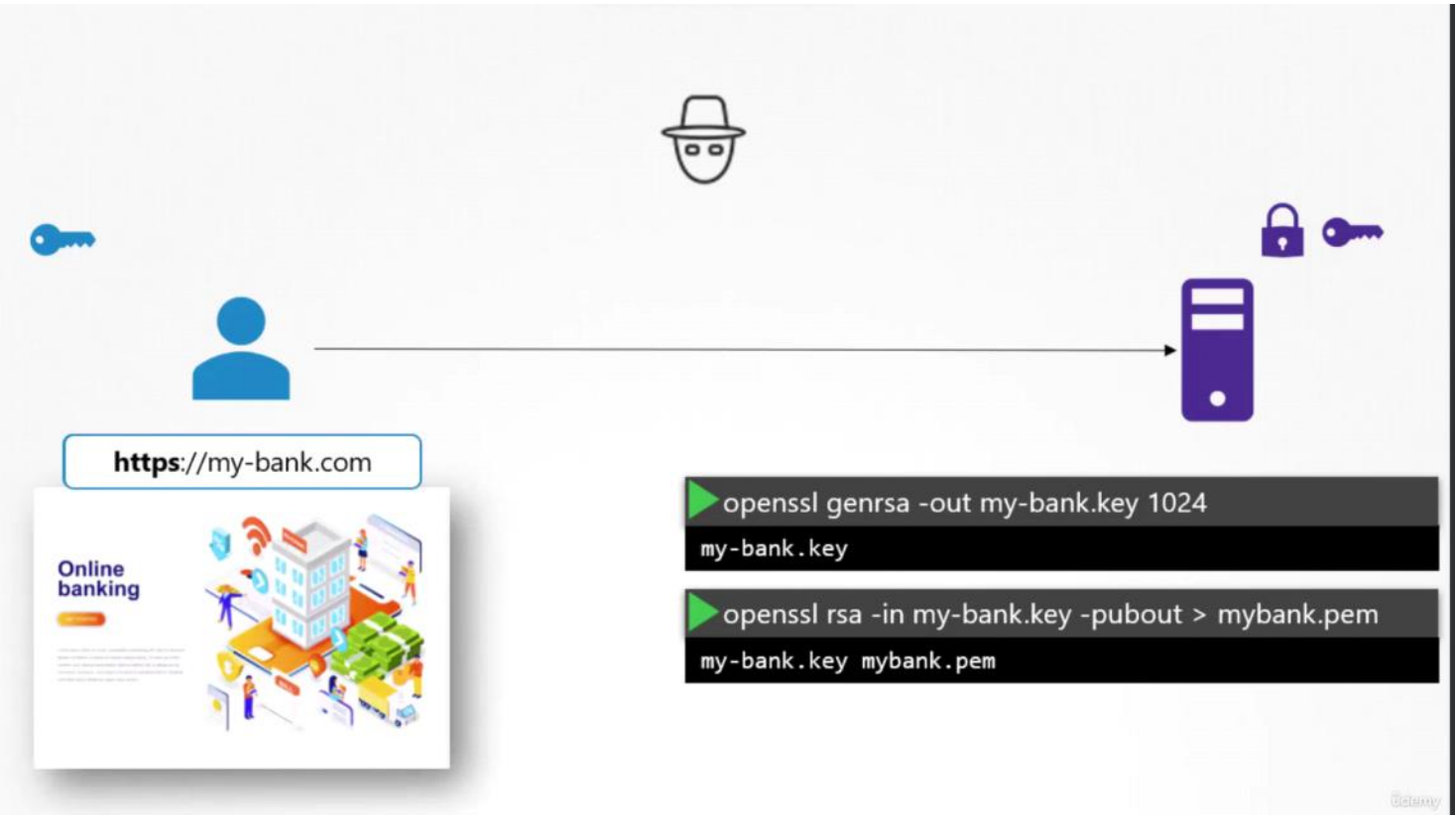
Friday, June 28, 2024 6:59 PM

Symetric encryption

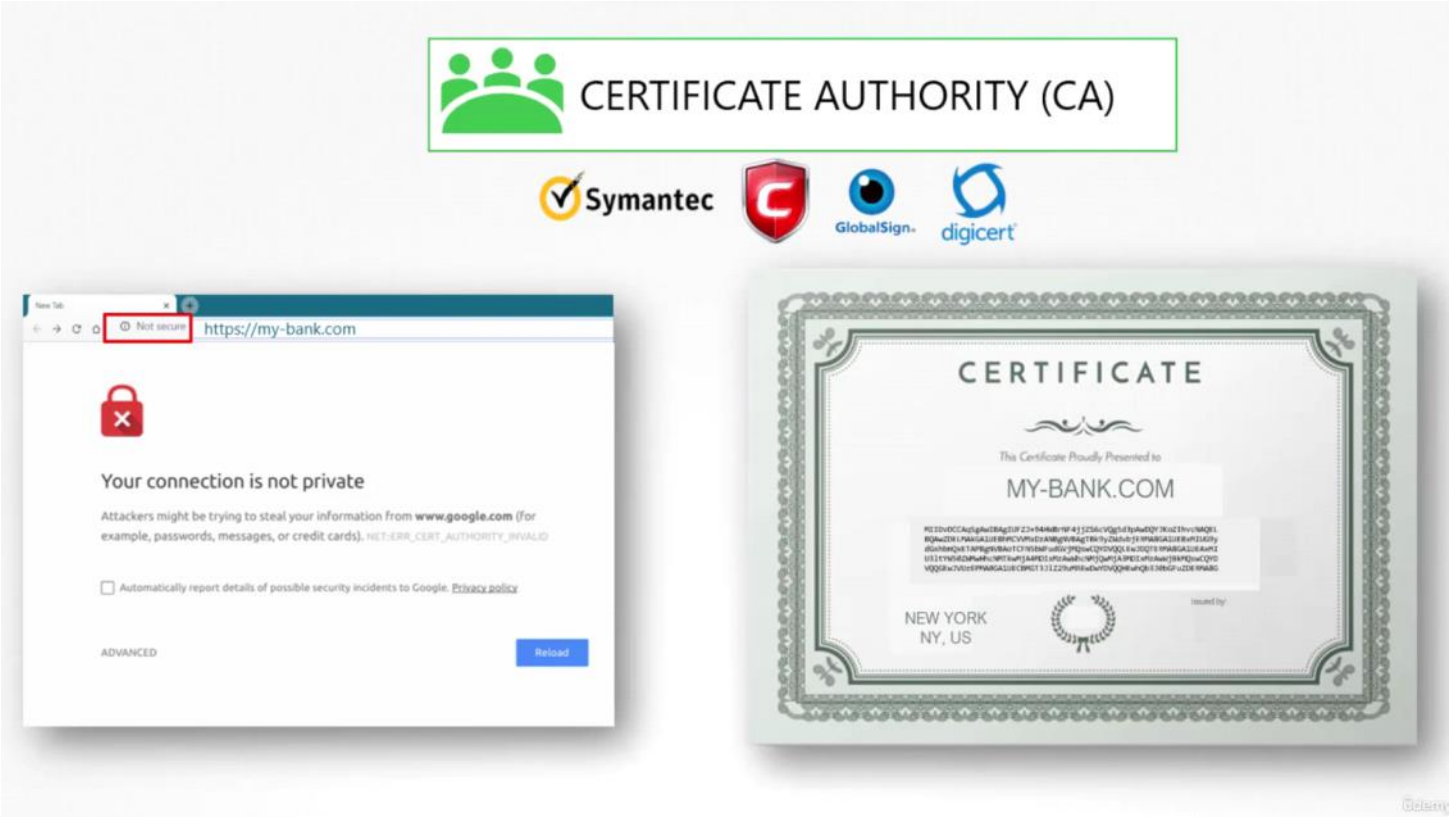


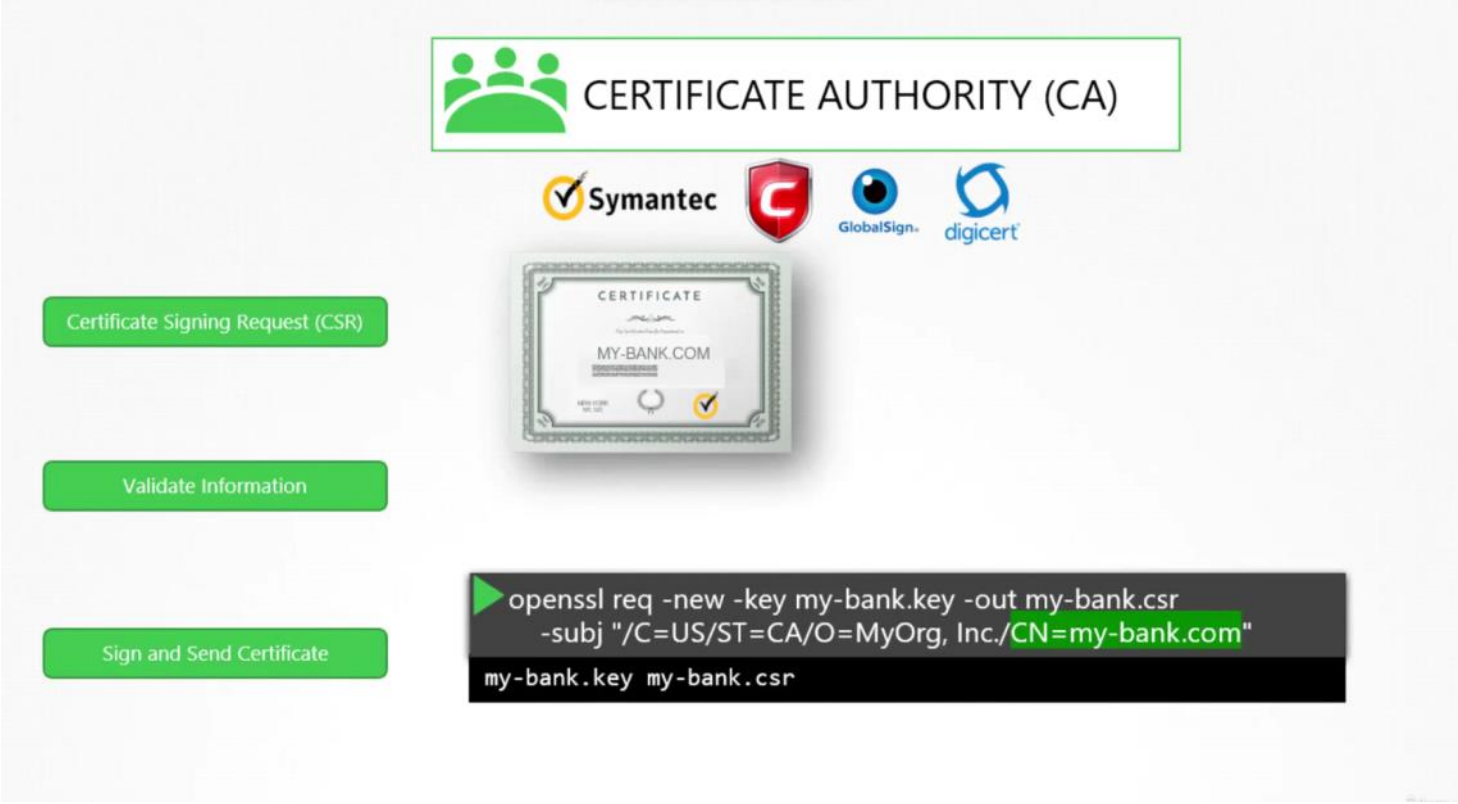

ASYMMETRIC ENCRYPTION - SSH
ssh-keygen
id_rsa id_rsa.pub
cat ~/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc…KhtUBfoTzlBqR
V1NThvOo4opzEwRQo1mWx user1
Private Key
Public Lock
ssh -i id_rsa user1@server1
Successfully Logged In!

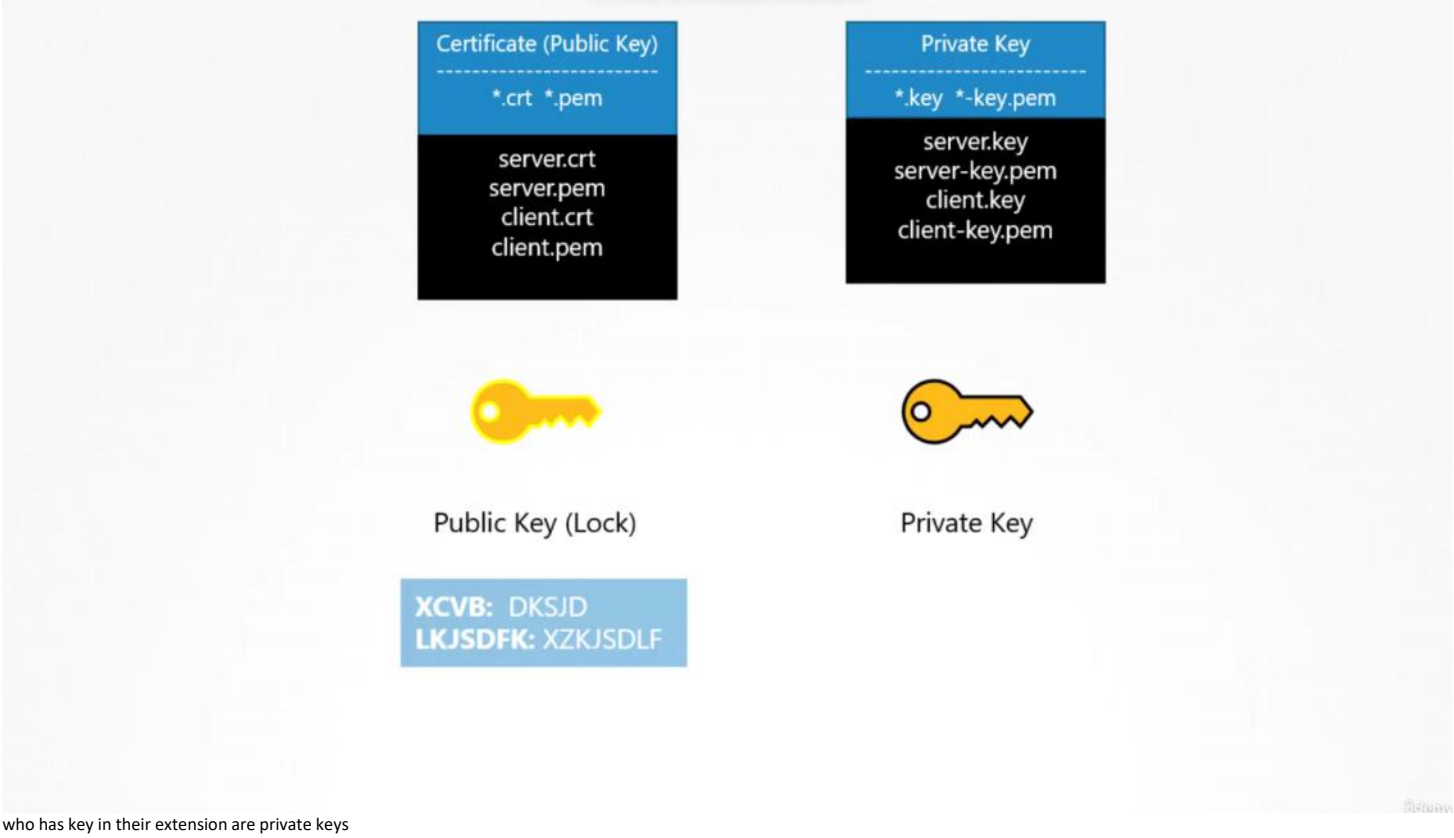

ASYMMETRIC ENCRYPTION - SSH
Private Key
Private Key
Public Lock
cat ~/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc…KhtUBfoTzlBqRV1NThvOo4opzEwRQo1mWx user1
ssh-rsa AAAXCVJSDFDF…SLKJSDLKFw23423xckjSDFDFLKJLSDFKJLx user2



generating ssl cert


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

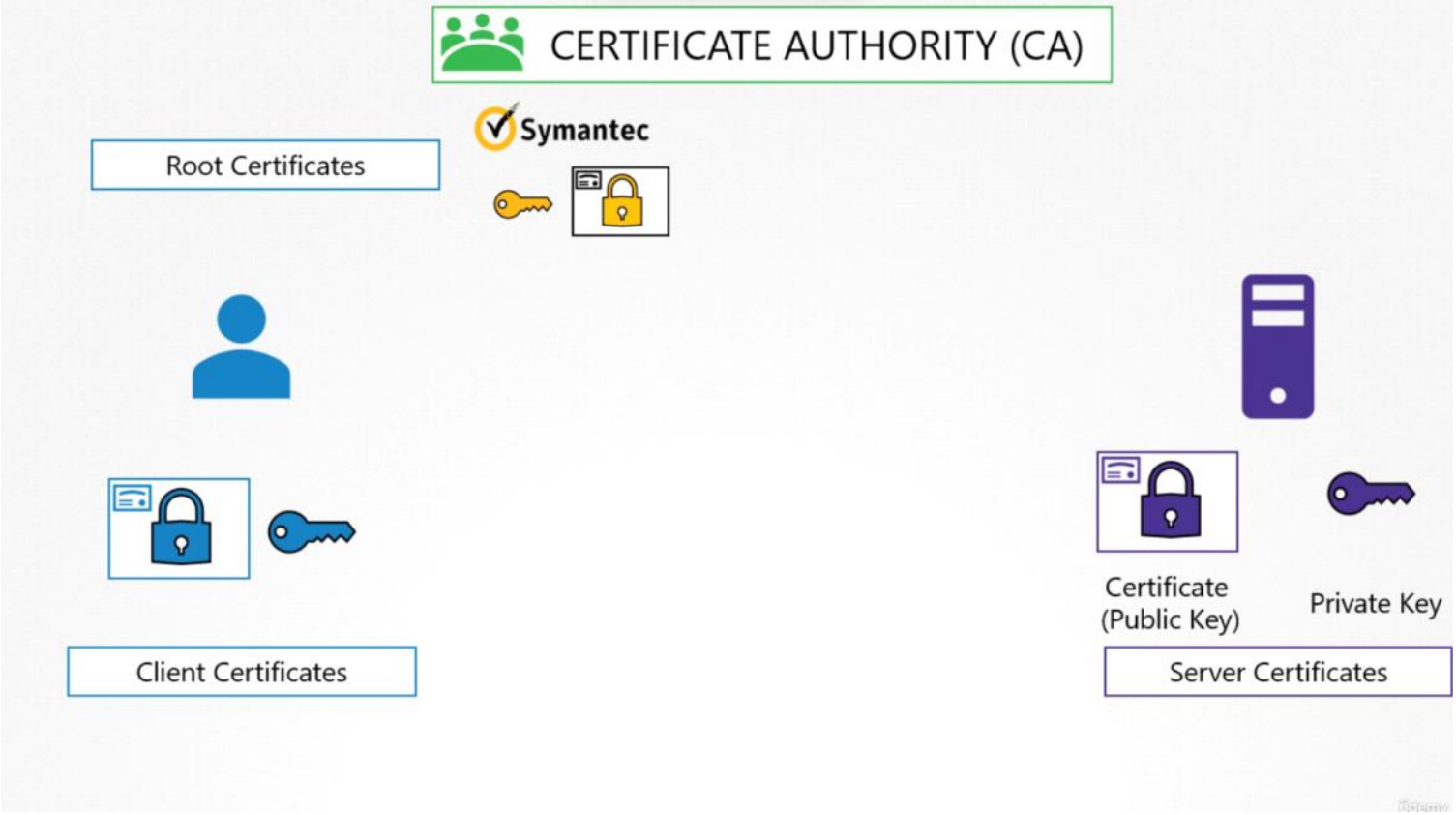

CERTIFICATE AUTHORITY (CA)
Symantec
GlobalSign
digicert
CERTIFICATE
MY-BANK.COM
Certificate Signing Request (CSR)
Validate Information
Sign and Send Certificate
openssl req -new -key my-bank.key -out my-bank.csr
-subj "/C=US/ST=CA/O=MyOrg, Inc./CN=my-bank.com"
my-bank.key my-bank.csr

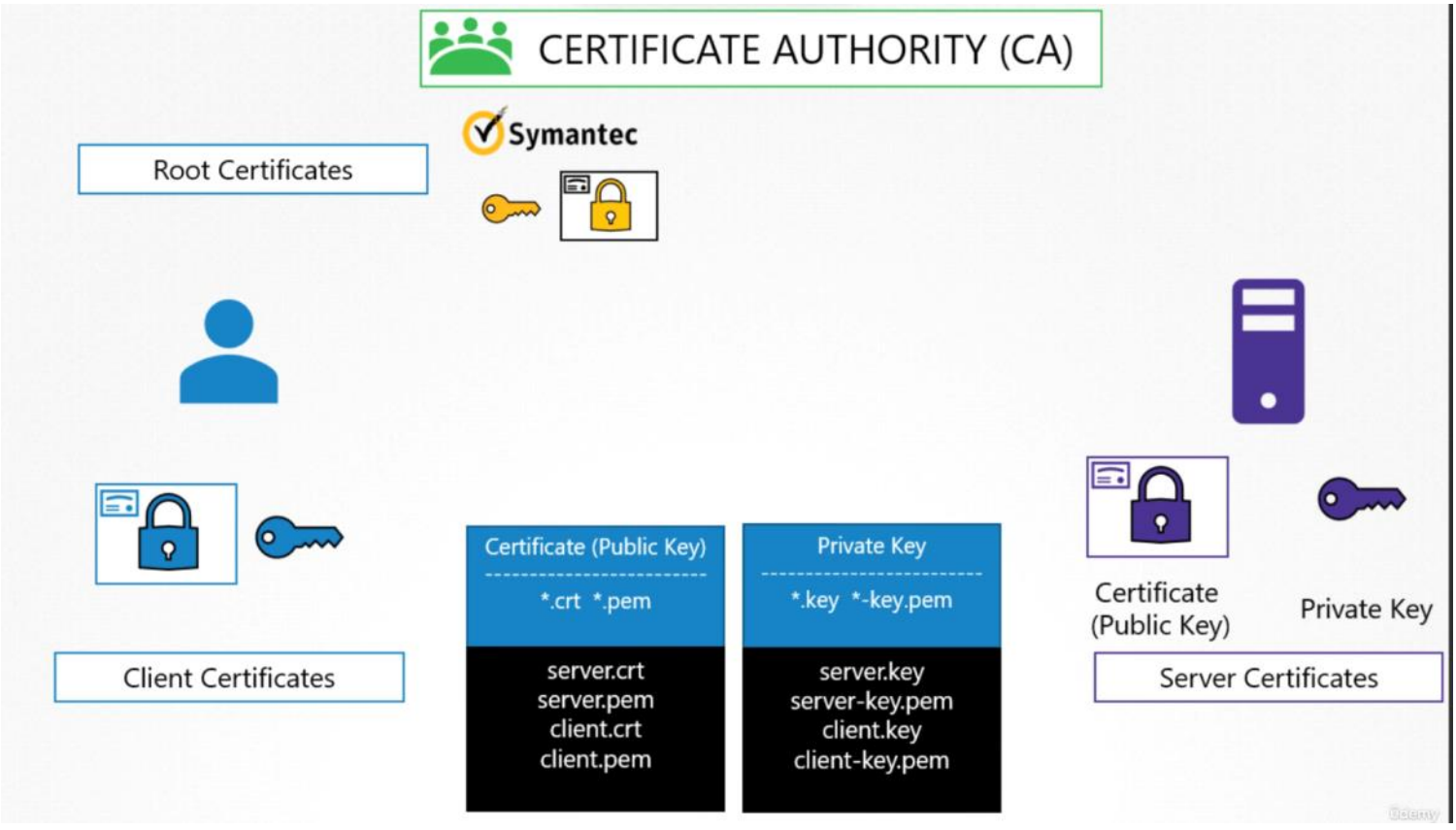


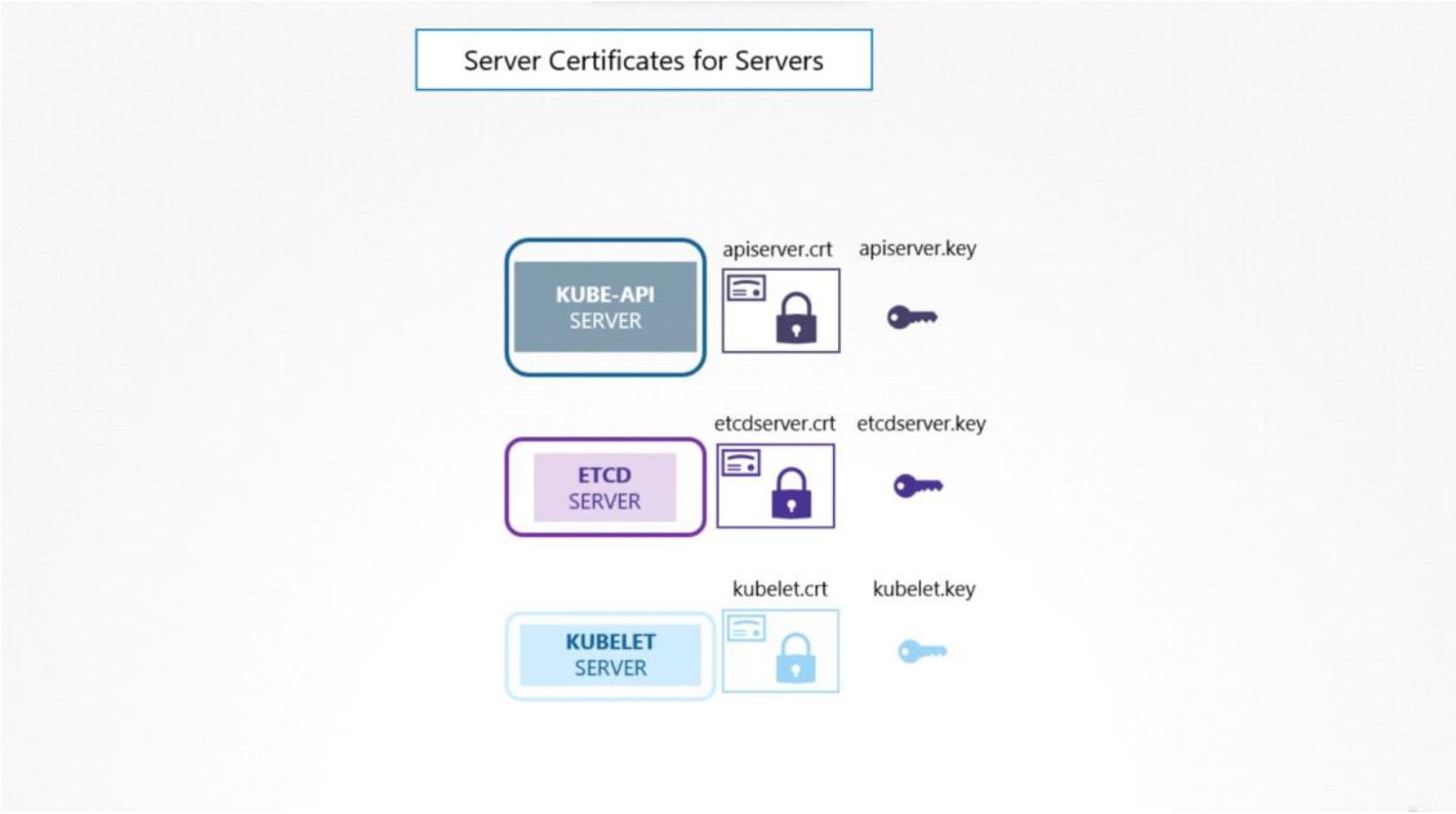
who has key in their extension are private keys

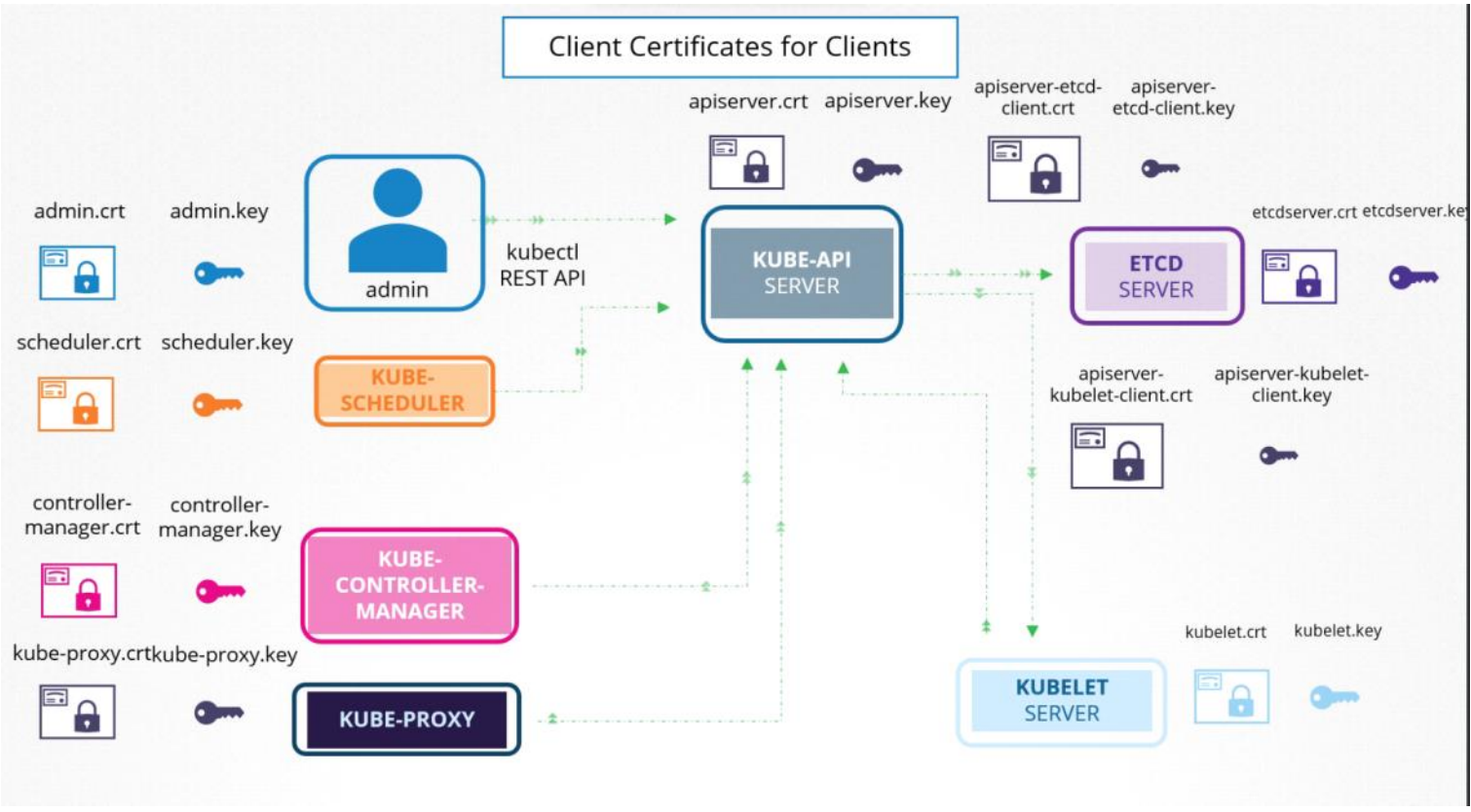
TLS in kubenetes

Friday, June 28, 2024 7:44 PM



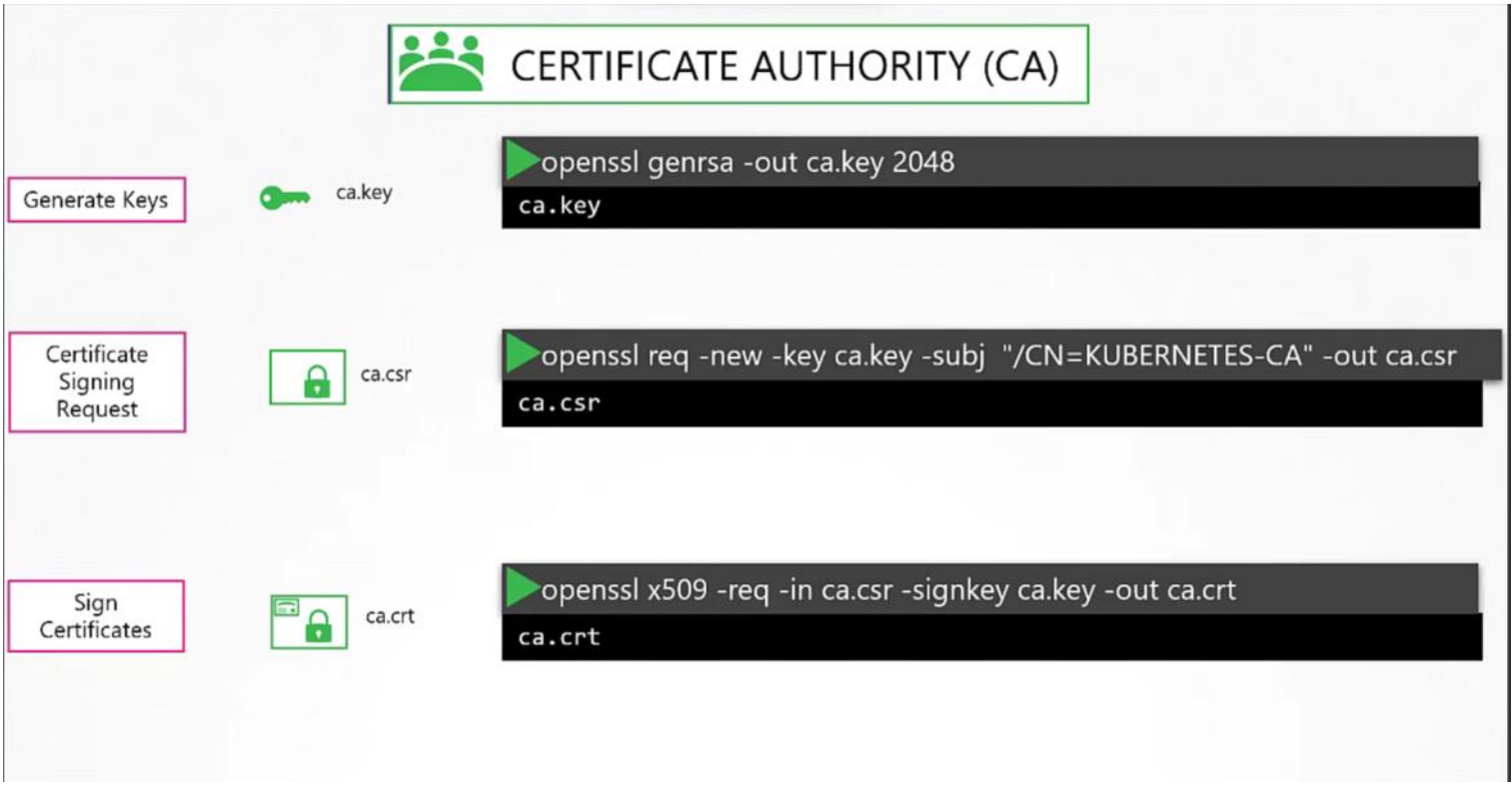


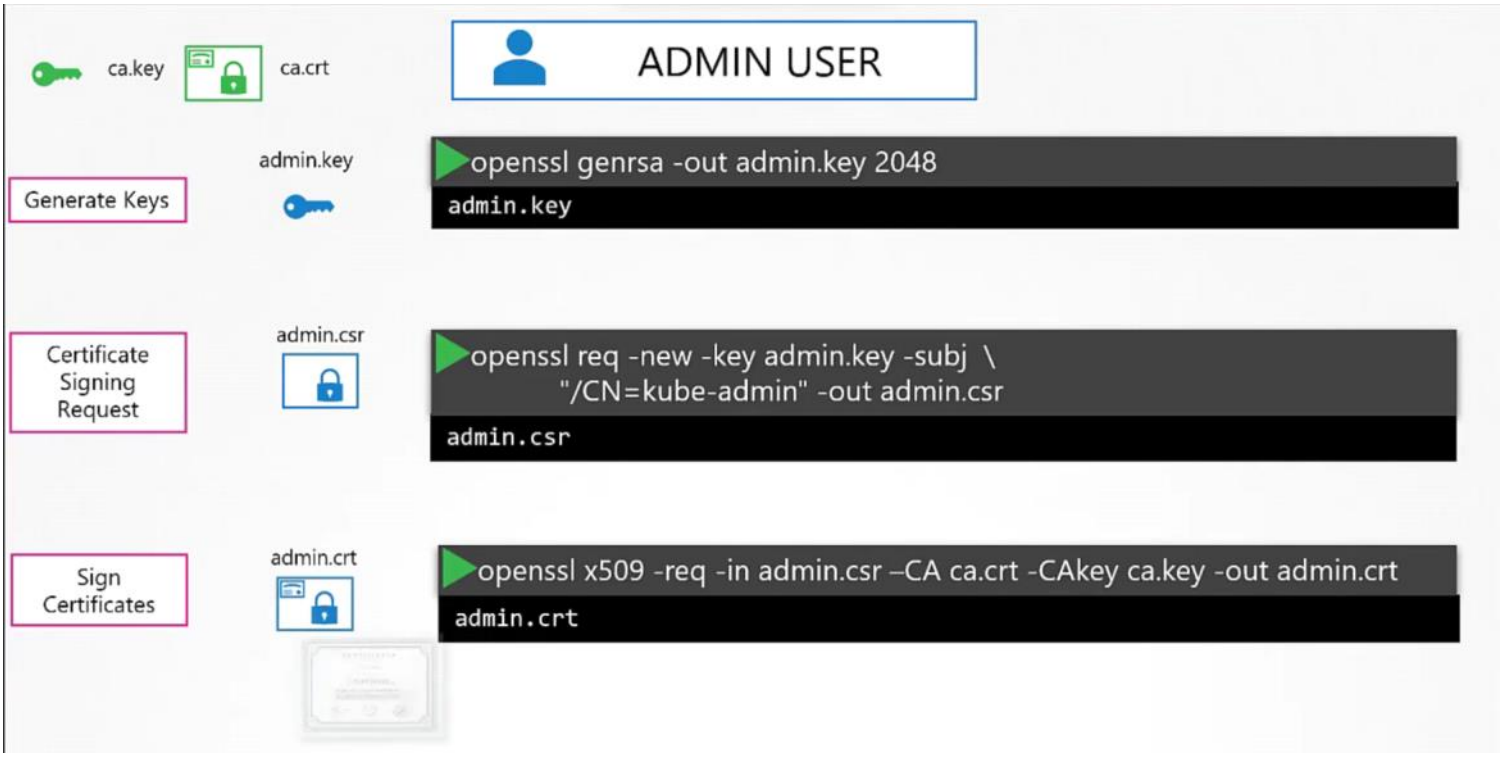

Server Certificates for Servers
KUBE-API SERVER
apiserver.crt
apiserver.key
ETCD SERVER
etcdserver.crt
etcdserver.key
KUBELET SERVER
kubelet.crt
kubelet.key

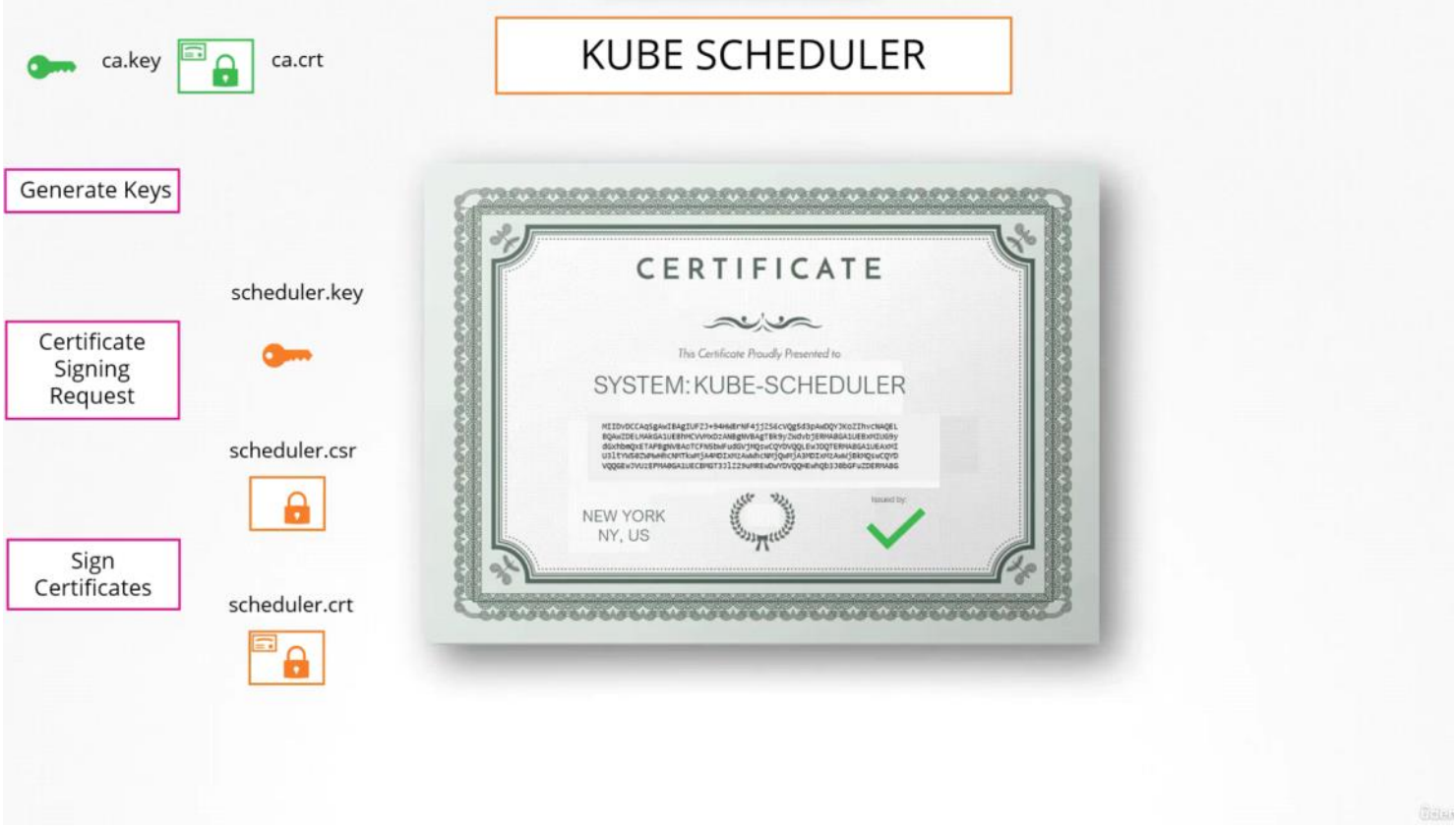

Client Certificates for Clients
admin.crt
admin.key
admin
kubectl REST API
apiserver.crt
apiserver.key
KUBE-API SERVER
apiserver-etcd-client.crt
apiserver-etcd-client.key
ETCD SERVER
etcdserver.crt
scheduler.crt
scheduler.key
KUBE-SCHEDULER
apiserver-kubelet-client.crt
apiserver-kubelet-client.key
controller-manager.crt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

Certificate creation

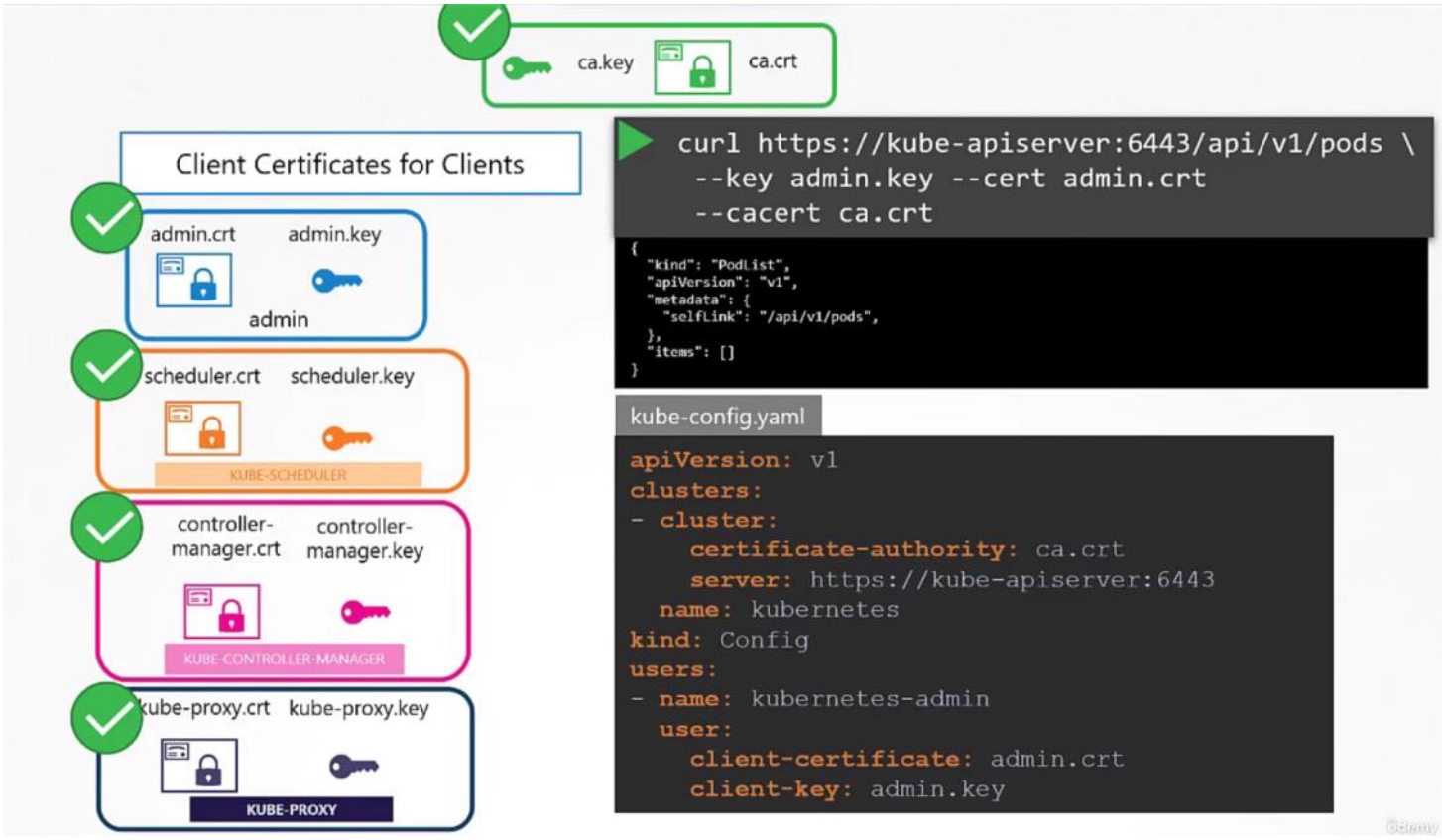
Friday, June 28, 2024 8:05 PM

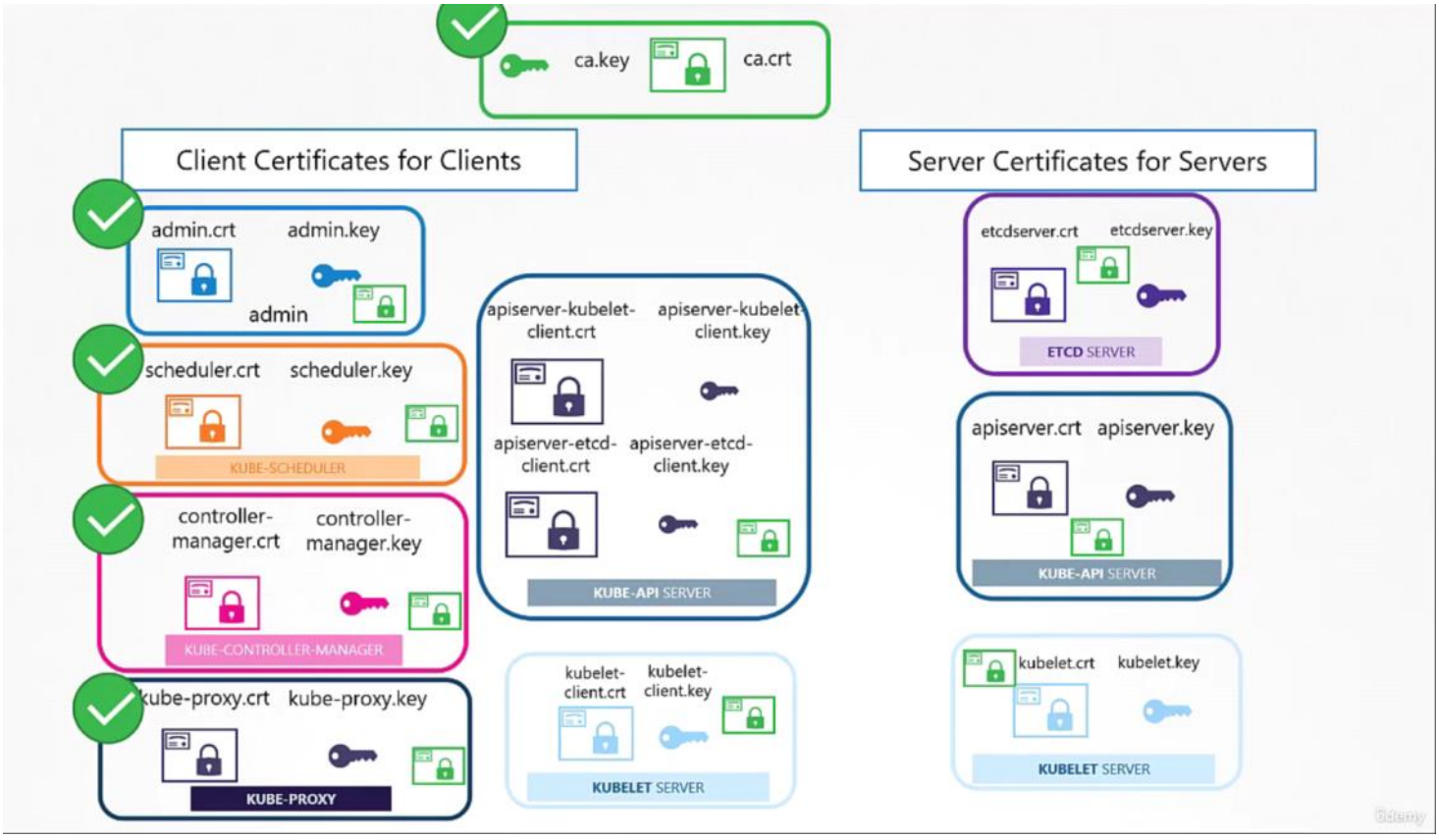



ca.key
ca.crt
ADMIN USER
admin.key
Generate Keys
openssl genrsa -out admin.key 2048
admin.key
admin.csr
Certificate Signing Request
openssl req -new -key admin.key -subj \
"/CN=kube-admin" -out admin.csr
admin.csr
admin.crt
Sign Certificates
openssl x509 -req -in admin.csr –CA ca.crt -CAkey ca.key -out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

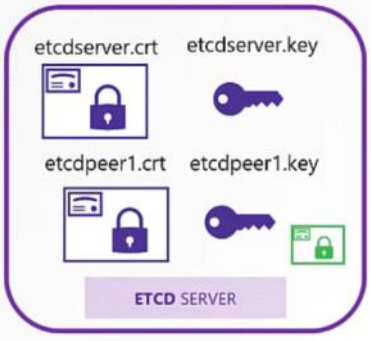

ca.key
ca.crt
KUBE SCHEDULER
Generate Keys
scheduler.key
Certificate Signing Request
scheduler.csr
Sign Certificates
scheduler.crt
CERTIFICATE
This Certificate Proudly Presented to
SYSTEM:KUBE-SCHEDULER
NEW YORK
NY, US
Issued by:

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


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



ETCD SERVERS




```
cat etcd.yaml
- etcd
  - --advertise-client-urls=https://127.0.0.1:2379
  - --key-file=/path-to-certs/etcdserver.key
  - --cert-file=/path-to-certs/etcdserver.crt
  - --client-cert-auth=true
  - --data-dir=/var/lib/etcd
  - --initial-advertise-peer-urls=https://127.0.0.1:2380
  - --initial-cluster=master=https://127.0.0.1:2380
  - --listen-client-urls=https://127.0.0.1:2379
  - --listen-peer-urls=https://127.0.0.1:2380
  - --name=master
  - --peer-cert-file=/path-to-certs/etcdpeer1.crt
  - --peer-client-cert-auth=true
  - --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
  - --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
  - --snapshot-count=10000
  - --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
```

KUBE API SERVER

```
openssl genrsa -out apiserver.key 2048
```

```
apiserver.key
```

```
openssl req -new -key apiserver.key -subj \
    "/CN=kube-apiserver" -out apiserver
```

KUBE API SERVER

```
openssl req -new -key apiserver.key -subj \
    "/CN=kube-apiserver" -out apiserver.csr -config openssl.cnf
```

```
apiserver.csr
```

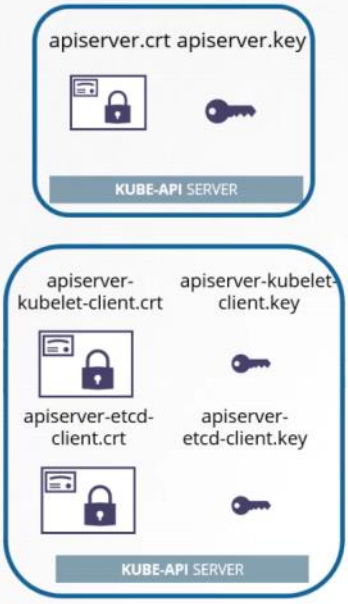
openssl.cnf

```
[req]
req_extensions = v3_req
distinguished_name = req_distinguished_name
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation,
subjectAltName = @alt_names
[alt_names]
DNS.1 = kubernetes
DNS.2 = kubernetes.default
DNS.3 = kubernetes.default.svc
DNS.4 = kubernetes.default.svc.cluster.local
IP.1 = 10.96.0.1
IP.2 = 172.17.0.87
```

```
openssl x509 -req -in apiserver.csr \
    -CA ca.crt -CAkey ca.key -out apiserver.crt
```

```
apiserver.crt
```

KUBE API SERVER

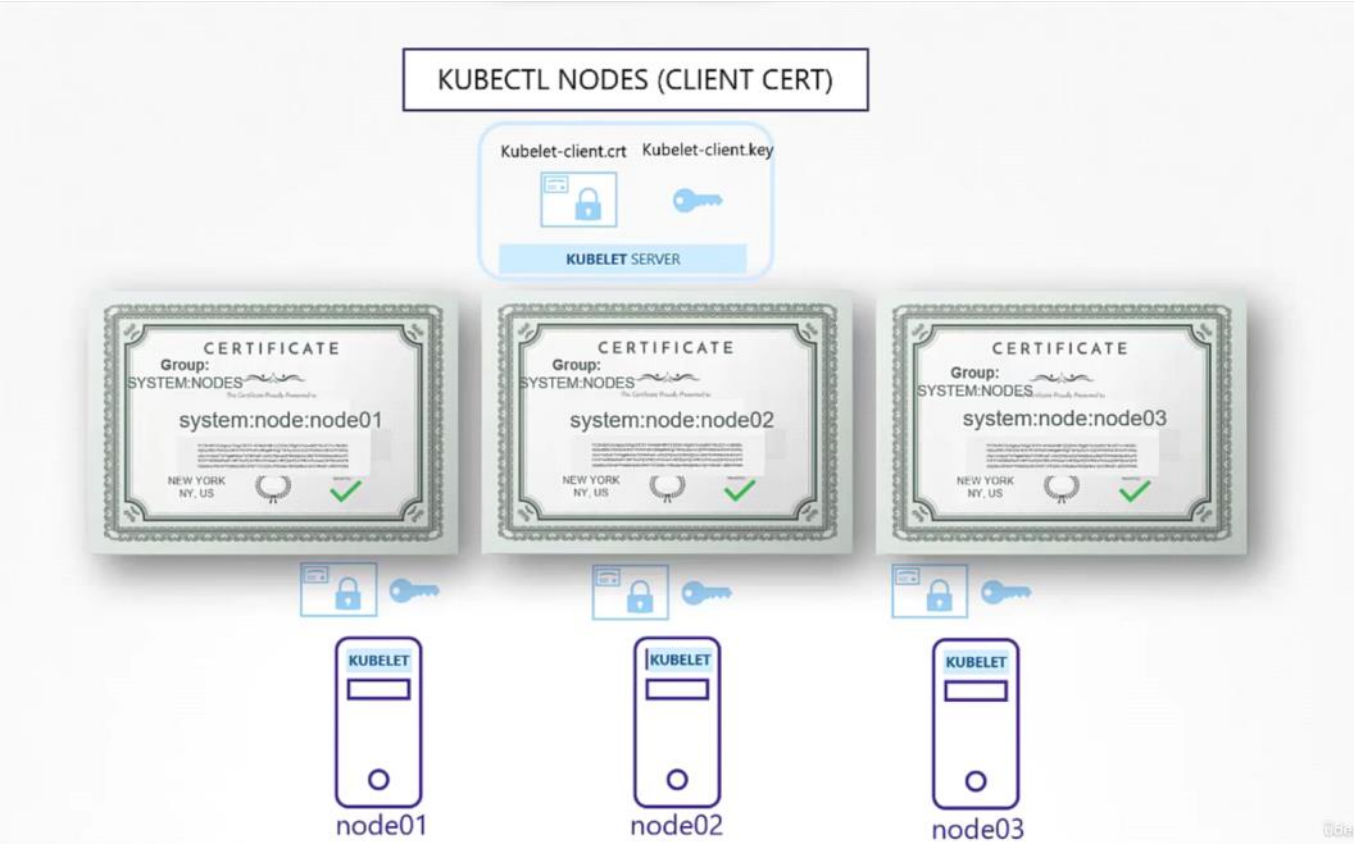



```
ExecStart=/usr/local/bin/kube-apiserver \\
  --advertise-address=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --authorization-mode=Node,RBAC \\
  --bind-address=0.0.0.0 \\
  --enable-swagger-ui=true \\
  --etcd-cafile=/var/lib/kubernetes/ca.pem \\
  --etcd-certfile=/var/lib/kubernetes/apiserver-etcd-client.crt \\
  --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-kubelet-client.crt \\
  --kubelet-client-key=/var/lib/kubernetes/apiserver-kubelet-client.key \\
  --kubelet-https=true \\
  --runtime-config=api/all \\
  --service-account-key-file=/var/lib/kubernetes/service-account.pem \\
  --service-cluster-ip-range=10.32.0.0/24 \\
  --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
```

KUBECTL NODES (SERVER CERT)

kubelet-config.yaml (node01)

```yaml
kind: KubeletConfiguration
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  x509:
    clientCAFile: "/var/lib/kubernetes/ca.pem"
authorization:
  mode: Webhook
clusterDomain: "cluster.local"
clusterDNS:
  - "10.32.0.10"
podCIDR: "${POD_CIDR}"
resolvConf: "/run/systemd/resolve/resolv.conf"
runtimeRequestTimeout: "15m"
tlsCertFile: "/var/lib/kubelet/kubelet-node01.crt"
tlsPrivateKeyFile: "/var/lib/kubelet/kubelet-node01.key"
```


KUBECTL NODES (CLIENT CERT)
Kubelet-client.crt
Kubelet-client.key
KUBELET SERVER
CERTIFICATE
Group:
SYSTEM:NODES
system:node:node01
NEW YORK
NY, US
CERTIFICATE
Group:
SYSTEM:NODES
system:node:node02
NEW YORK
NY, US
CERTIFICATE
Group:
SYSTEM:NODES
system:node:node03
NEW YORK
NY, US
KUBELET
node01
KUBELET
node02
KUBELET
node03

View certificate details

Friday, June 28, 2024 8:22 PM

```
cat /etc/kubernetes/manifests/kube-apiserver.yaml
spec:
  containers:
  - command:
    - kube-apiserver
    - --authorization-mode=Node,RBAC
    - --advertise-address=172.17.0.32
    - --allow-privileged=true
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --disable-admission-plugins=PersistentVolumeLabel
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
    - --secure-port=6443
    - --service-account-key-file=/etc/kubernetes/pki/sa.pub
    - --service-cluster-ip-range=10.96.0.0/12
    - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
    - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
```

Practice- view cert detail

Friday, June 28, 2024 8:27 PM

Task Hint 59:11

01 02 03 04 05 06 07 08 09 10 11 12 13

Identify the certificate file used for the `kube-api server`

/etc/kubernetes/pki/apiserver.crt

/etc/apiserver.crt

/etc/apiserver.key

/tmp/kube-apiserver.crt

/etc/kubernetes/pki/kube-apiserver.crt

Terminal 1

```
root@controlplane:~#
root@controlplane:~# cat /etc/kubernetes/manifests/
etcd.yaml                    kube-controller-manager.yaml
kube-apiserver.yaml          kube-scheduler.yaml
root@controlplane:~# cat /etc/kubernetes/manifests/kube-apiserver.yaml
```

Task Hint 58:38

01 02 03 04 05 06 07 08 09 10 11 12 13

Identify the certificate file used for the `kube-api server`

/etc/kubernetes/pki/apiserver.crt

/etc/apiserver.crt

/etc/apiserver.key

/tmp/kube-apiserver.crt

/etc/kubernetes/pki/kube-apiserver.crt

Terminal 1

```
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=10.46.98.9
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
    - --requestheader-allowed-names=front-proxy-client
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
    - --requestheader-extra-headers-prefix=X-Remote-Extra-
    - --requestheader-group-headers=X-Remote-Group
    - --requestheader-username-headers=X-Remote-User
    - --secure-port=6443
    - --service-account-issuer=https://kubernetes.default.svc.cluster.local
    - --service-account-key-file=/etc/kubernetes/pki/sa.pub
    - --service-account-signing-key-file=/etc/kubernetes/pki/sa.key
    - --service-cluster-ip-range=10.96.0.0/12
    - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
    - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
    image: k8s.gcr.io/kube-apiserver:v1.20.0
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 8
      httpGet:
        host: 10.46.98.9
```

Task Hint 57:42

01 02 03 04 05 06 07 08 09 10 11 12 13

Identify the Certificate file used to authenticate `kube-apiserver` as a client to `ETCD` Server

/etc/kubernetes/pki/apiserver-etcd-client.key

/etc/kubernetes/pki/apiserver.crt

/etc/kubernetes/pki/apiserver-kubelet-client.crt

/etc/kubernetes/pki/apiserver-etcd-client.crt

/etc/kubernetes/pki/apiserver-etcd-client.crt

Terminal 1

```
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=10.46.98.9
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
    - --requestheader-allowed-names=front-proxy-client
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
    - --requestheader-extra-headers-prefix=X-Remote-Extra-
    - --requestheader-group-headers=X-Remote-Group
    - --requestheader-username-headers=X-Remote-User
    - --secure-port=6443
    - --service-account-issuer=https://kubernetes.default.svc.cluster.local
    - --service-account-key-file=/etc/kubernetes/pki/sa.pub
    - --service-account-signing-key-file=/etc/kubernetes/pki/sa.key
    - --service-cluster-ip-range=10.96.0.0/12
    - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
    - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
    image: k8s.gcr.io/kube-apiserver:v1.20.0
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 8
      httpGet:
        host: 10.46.98.9
```

Task Hint 56:57

01 02 03 04 05 06 07 08 09 10 11 12 13

Identify the key used to authenticate `kubeapi-server` to the `kubelet` server

/etc/kubernetes/pki/apiserver-kubelet-client.crt

/etc/kubernetes/pki/front-proxy-client.key

/etc/kubernetes/pki/apiserver-kubelet-client.key

/etc/kubernetes/pki/apiserver.key

/etc/kubernetes/pki/apiserver-etcd-client.key

Terminal 1

```
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=10.46.98.9
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
    - --requestheader-allowed-names=front-proxy-client
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
    - --requestheader-extra-headers-prefix=X-Remote-Extra-
    - --requestheader-group-headers=X-Remote-Group
    - --requestheader-username-headers=X-Remote-User
    - --secure-port=6443
    - --service-account-issuer=https://kubernetes.default.svc.cluster.local
    - --service-account-key-file=/etc/kubernetes/pki/sa.pub
    - --service-account-signing-key-file=/etc/kubernetes/pki/sa.key
    - --service-cluster-ip-range=10.96.0.0/12
    - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
    - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
    image: k8s.gcr.io/kube-apiserver:v1.20.0
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 8
      httpGet:
        host: 10.46.98.9
```

Task Hint 55:30

01 02 03 04 05 06 07 08 09 10 11 12 13

Identify the ETCD Server Certificate used to host ETCD server

/etc/kubernetes/pki/apiserver-etcd-client.crt

/etc/kubernetes/pki/etcd/ca.crt

/etc/kubernetes/pki/etcd/server.crt

/etc/kubernetes/pki/apiserver.crt

Terminal 1 +

```
    - mountPath: /etc/ca-certificates
      name: etc-ca-certificates
      readOnly: true
    - mountPath: /etc/kubernetes/pki
      name: k8s-certs
      readOnly: true
    - mountPath: /usr/local/share/ca-certificates
      name: usr-local-share-ca-certificates
      readOnly: true
    - mountPath: /usr/share/ca-certificates
      name: usr-share-ca-certificates
      readOnly: true
  hostNetwork: true
  priorityClassName: system-node-critical
  volumes:
  - hostPath:
      path: /etc/ssl/certs
      type: DirectoryOrCreate
    name: ca-certs
  - hostPath:
      path: /etc/ca-certificates
      type: DirectoryOrCreate
    name: etc-ca-certificates
  - hostPath:
      path: /etc/kubernetes/pki
      type: DirectoryOrCreate
    name: k8s-certs
  - hostPath:
      path: /usr/local/share/ca-certificates
      type: DirectoryOrCreate
    name: usr-local-share-ca-certificates
  - hostPath:
      path: /usr/share/ca-certificates
      type: DirectoryOrCreate
    name: usr-share-ca-certificates
status: {}
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# cat /etc/kubernetes/manifests/etcd.yaml
```

Task Hint 54:38

01 02 03 04 05 06 07 08 09 10 11 12 13

Identify the ETCD Server Certificate used to host ETCD server

/etc/kubernetes/pki/apiserver-etcd-client.crt

/etc/kubernetes/pki/etcd/ca.crt

/etc/kubernetes/pki/etcd/server.crt

/etc/kubernetes/pki/apiserver.crt

Terminal 1 +

```
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/etcd.advertise-client-urls: https://10.46.98.9:2379
  creationTimestamp: null
  labels:
    component: etcd
    tier: control-plane
  name: etcd
  namespace: kube-system
spec:
  containers:
  - command:
    - etcd
    - --advertise-client-urls=https://10.46.98.9:2379
    - --cert-file=/etc/kubernetes/pki/etcd/server.crt
    - --client-cert-auth=true
    - --data-dir=/var/lib/etcd
    - --initial-advertise-peer-urls=https://10.46.98.9:2380
    - --initial-cluster=controlplane=https://10.46.98.9:2380
    - --key-file=/etc/kubernetes/pki/etcd/server.key
    - --listen-client-urls=https://127.0.0.1:2379,https://10.46.98.9:2379
    - --listen-metrics-urls=http://127.0.0.1:2381
    - --listen-peer-urls=https://10.46.98.9:2380
    - --name=controlplane
    - --peer-cert-file=/etc/kubernetes/pki/etcd/peer.crt
    - --peer-client-cert-auth=true
    - --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
    - --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
    - --snapshot-count=10000
    - --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
    image: k8s.gcr.io/etcd:3.4.13-0
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 8
      httpGet:
        host: 127.0.0.1
        path: /health
        port: 2381
```

Task Hint 54:00

01 02 03 04 05 06 07 08 09 10 11 12 13

Identify the ETCD Server CA Root Certificate used to serve ETCD Server

ETCD can have its own CA. So this may be a different CA certificate than the one used by kube-api server.

/etc/kubernetes/pki/etcd/ca.crt

/etc/kubernetes/pki/etcd/server.crt

/etc/kubernetes/pki/ca.crt

/etc/kubernetes/pki/apiserver-kubelet-client.crt

Terminal 1 +

```
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/etcd.advertise-client-urls: https://10.46.98.9:2379
  creationTimestamp: null
  labels:
    component: etcd
    tier: control-plane
  name: etcd
  namespace: kube-system
spec:
  containers:
  - command:
    - etcd
    - --advertise-client-urls=https://10.46.98.9:2379
    - --cert-file=/etc/kubernetes/pki/etcd/server.crt
    - --client-cert-auth=true
    - --data-dir=/var/lib/etcd
    - --initial-advertise-peer-urls=https://10.46.98.9:2380
    - --initial-cluster=controlplane=https://10.46.98.9:2380
    - --key-file=/etc/kubernetes/pki/etcd/server.key
    - --listen-client-urls=https://127.0.0.1:2379,https://10.46.98.9:2379
    - --listen-metrics-urls=http://127.0.0.1:2381
    - --listen-peer-urls=https://10.46.98.9:2380
    - --name=controlplane
    - --peer-cert-file=/etc/kubernetes/pki/etcd/peer.crt
    - --peer-client-cert-auth=true
    - --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
    - --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
    - --snapshot-count=10000
    - --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
    image: k8s.gcr.io/etcd:3.4.13-0
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 8
      httpGet:
        host: 127.0.0.1
        path: /health
        port: 2381
```

the kube api server has its own ca file and etcd server has its own ca file.

01 02 03 04 05 06 07 08 09 10 11 12 13

What is the Common Name (CN) configured on the Kube API Server Certificate?

OpenSSL Syntax: `openssl x509 -in file-path.crt -text -noout`

kube-apiserver

kubernetes

kube-api-server

api-server

kubeapi-server

```
  hostNetwork: true
  priorityClassName: system-node-critical
  volumes:
  - hostPath:
      path: /etc/kubernetes/pki/etcd
      type: DirectoryOrCreate
    name: etcd-certs
  - hostPath:
      path: /var/lib/etcd
      type: DirectoryOrCreate
    name: etcd-data
status: {}
root@controlplane:~# openssl x509 -in /etc/kubernetes/pki/apiserver.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 231788572510502865 (0x3377a7a877bbfd1)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = kubernetes
        Validity
            Not Before: Apr 17 21:17:20 2022 GMT
            Not After : Apr 17 21:17:20 2023 GMT
        Subject: CN = kube-apiserver
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b4:e6:15:51:7c:c9:9e:78:5d:3e:09:85:5b:bf:
                    03:78:d5:4a:44:e7:54:b4:e0:57:7a:28:7b:37:1b:
                    93:49:a7:91:7a:70:3b:bc:fa:28:c6:e4:38:d8:17:
                    b9:98:f7:91:70:e4:64:d8:8a:09:4c:29:00:17:1f:
                    70:01:ef:64:17:25:91:8c:1f:95:aa:f1:68:84:2f:
                    7a:96:45:dc:e8:e8:c2:f7:73:fa:6e:41:b4:ef:89:
                    cd:7e:25:bb:79:60:86:cb:96:96:f3:c7:d7:16:a4:
                    b5:ba:af:27:b8:3e:51:55:11:c7:97:48:c5:f4:89:
                    45:28:d9:0b:b5:f5:fd:13:74:1d:fe:48:02:54:ad:
                    5c:cf:cb:fa:45:cb:5e:47:63:10:7f:7d:45:85:4f:
                    ac:36:50:80:ab:bf:08:cb:cc:ea:53:97:8d:ca:79:
                    3a:67:ed:f7:d7:b4:c7:68:83:34:64:d1:e0:5b:6c:
```

openssl x509 -in /etc/kubernetes/pki/apiserver.crt -text

01 02 03 04 05 06 07 08 09 10 11 12 13

What is the name of the CA who issued the Kube API Server Certificate?

kubernetes-ca

kube-apiserver

ca

kubernetes

```
  hostNetwork: true
  priorityClassName: system-node-critical
  volumes:
  - hostPath:
      path: /etc/kubernetes/pki/etcd
      type: DirectoryOrCreate
    name: etcd-certs
  - hostPath:
      path: /var/lib/etcd
      type: DirectoryOrCreate
    name: etcd-data
status: {}
root@controlplane:~# openssl x509 -in /etc/kubernetes/pki/apiserver.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 231788572510502865 (0x3377a7a877bbfd1)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = kubernetes
        Validity
            Not Before: Apr 17 21:17:20 2022 GMT
            Not After : Apr 17 21:17:20 2023 GMT
        Subject: CN = kube-apiserver
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b4:e6:15:51:7c:c9:9e:78:5d:3e:09:85:5b:bf:
                    03:78:d5:4a:44:e7:54:b4:e0:57:7a:28:7b:37:1b:
                    93:49:a7:91:7a:70:3b:bc:fa:28:c6:e4:38:d8:17:
                    b9:98:f7:91:70:e4:64:d8:8a:09:4c:29:00:17:1f:
                    70:01:ef:64:17:25:91:8c:1f:95:aa:f1:68:84:2f:
                    7a:96:45:dc:e8:e8:c2:f7:73:fa:6e:41:b4:ef:89:
                    cd:7e:25:bb:79:60:86:cb:96:96:f3:c7:d7:16:a4:
                    b5:ba:af:27:b8:3e:51:55:11:c7:97:48:c5:f4:89:
                    45:28:d9:0b:b5:f5:fd:13:74:1d:fe:48:02:54:ad:
                    5c:cf:cb:fa:45:cb:5e:47:63:10:7f:7d:45:85:4f:
                    ac:36:50:80:ab:bf:08:cb:cc:ea:53:97:8d:ca:79:
                    3a:67:ed:f7:d7:b4:c7:68:83:34:64:d1:e0:5b:6c:
```

01 02 03 04 05 06 07 08 09 10 11 12 13

Which of the below alternate names is not configured on the Kube API Server Certificate?

kubernetes

controlplane

kubernetes.default.svc

kube-master

```
                    7a:96:45:dc:e8:e8:c2:f7:73:fa:6e:41:b4:ef:89:
                    cd:7e:25:bb:79:60:86:cb:96:96:f3:c7:d7:16:a4:
                    b5:ba:af:27:b8:3e:51:55:11:c7:97:48:c5:f4:89:
                    45:28:d9:0b:b5:f5:fd:13:74:1d:fe:48:02:54:ad:
                    5c:cf:cb:fa:45:cb:5e:47:63:10:7f:7d:45:85:4f:
                    ac:36:50:80:ab:bf:08:cb:cc:ea:53:97:8d:ca:79:
                    3a:67:ed:f7:d7:b4:c7:68:83:34:64:d1:e0:5b:6c:
                    3d:80:c1:5c:5f:a4:c2:ab:03:7f:be:8a:84:7d:30:
                    71:61:bd:37:86:94:99:25:4c:b1:fc:f4:60:56:36:
                    60:d5:d2:be:dd:74:8a:39:6d:84:35:47:5c:08:31:
                    5f:df:ae:e6:c4:08:4a:8c:5d:c2:e1:db:13:45:44:
                    79:cd:55:21:21:1e:20:bf:15:50:f5:b0:4c:97:ab:
                    ae:bd
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Authority Key Identifier:
                keyid:F8:CB:6C:0F:51:70:60:42:1A:61:18:16:88:99:3E:DF:0E:1C:63:F7

            X509v3 Subject Alternative Name:
                DNS:controlplane, DNS:kubernetes, DNS:kubernetes.default, DNS:kubernetes.default.svc, DN
S:kubernetes.default.svc.cluster.local, IP Address:10.96.0.1, IP Address:10.46.98.9
    Signature Algorithm: sha256WithRSAEncryption
        60:4c:2e:78:d2:69:7e:b1:7b:6e:1b:ad:a3:1c:58:43:72:75:
        98:e3:46:bb:e8:38:70:47:92:70:52:73:ab:ea:e8:0d:66:42:
        92:0d:48:52:18:37:b6:f1:40:44:75:64:90:57:03:1d:53:3b:
        e7:e8:46:00:ff:78:09:c0:46:10:6a:9c:82:92:50:7a:40:00:
        cb:ab:b9:a4:90:c6:f7:e9:12:5b:f7:a2:ab:d6:e7:be:d7:a3:
        ae:5e:86:63:56:07:c9:b0:dd:80:a9:22:75:f3:91:7a:14:43:
        2e:1a:8a:ca:44:86:7f:69:10:9b:20:77:c7:a9:ad:34:a8:6c:
        6a:fa:c9:77:64:b7:fc:9e:f0:2b:60:a0:c5:37:88:b0:69:f0:
        52:fb:bc:5b:44:ff:04:ec:f2:cd:5a:72:78:6f:93:5e:38:27:
        4f:42:10:dd:99:da:74:78:e8:98:50:cb:f5:21:d2:95:17:fe:
        c3:df:ff:19:5f:90:50:5f:2b:3e:3d:b7:33:b8:b9:5b:1f:3c:
        ed:be:40:24:9e:93:ba:b0:36:d8:e1:a8:d1:b2:1c:32:5d:d6:
        7a:bc:4d:13:70:06:e5:67:ad:3b:ea:7b:26:5e:d2:15:59:2d:
```

`

01 02 03 04 05 06 07 08 09 10 11 12 13

What is the Common Name (CN) configured on the ETCD Server certificate?

kubernetes

controlplane

etcd-server

etcd

```
         c3:df:ff:19:5f:90:50:5f:2b:3e:3d:b7:33:b8:b9:5b:1f:3c:
         ed:be:40:24:9e:93:ba:b0:36:d8:e1:a8:d1:b2:1c:32:5d:d6:
         7a:bc:4d:13:70:06:e5:67:ad:3b:ea:7b:26:5e:d2:15:59:2d:
         74:7c:d7:90:5c:dc:f6:2d:12:07:24:3a:ce:fc:f1:b9:92:23:
         2d:5f:35:04
root@controlplane:~# openssl x509 -in /etc/kubernetes/pki/etcd/server.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 809626102485369927 (0xb3c5ea68f864447)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = etcd-ca
        Validity
            Not Before: Apr 17 21:17:21 2022 GMT
            Not After : Apr 17 21:17:22 2023 GMT
        Subject: CN = controlplane
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:c6:49:9b:5b:a8:1e:78:d3:e0:10:d9:3a:fc:8c:
                    ae:d0:c0:04:e2:26:00:83:15:ec:20:56:02:76:93:
                    f2:8a:a7:e7:34:e7:47:0b:e2:34:1b:40:f8:e9:b3:
                    0a:fa:fc:78:00:f6:a8:b7:12:10:95:59:e4:3e:32:
                    14:44:84:c0:5e:0e:98:4e:2c:96:c7:1b:b2:a8:3c:
                    a3:a9:c6:22:57:15:a7:c5:94:cd:00:a6:d0:e0:c2:
                    e8:86:25:ae:19:81:1a:c0:e5:8d:69:d9:61:51:45:
                    2d:21:c7:e3:61:6a:41:1f:d6:5f:28:87:39:f1:a2:
                    ab:6d:3f:f6:25:c5:1b:33:b6:fb:0f:78:51:d6:24:
                    68:72:07:a8:97:af:99:96:71:62:a3:e6:c8:58:c7:
                    f8:c5:ea:19:07:5f:ae:a0:2d:ed:9e:2f:d6:2e:db:
                    3d:47:8e:08:f8:7a:6f:f8:88:da:2e:89:99:e0:97:
                    a2:62:9a:b1:4c:a2:94:30:a5:14:7b:9b:16:14:c1:
                    52:dd:16:b4:5d:53:39:e5:9b:f2:aa:0b:ed:d6:08:
                    f1:d3:54:e9:57:fe:2f:b5:33:7d:c0:84:33:1b:8b:
                    fd:cf:8f:0c:e5:d8:93:3f:1d:06:9b:84:cf:db:b8:
                    51:58:c9:43:21:b1:7d:03:8b:24:1d:aa:af:05:c5:
                    d3:ad
                Exponent: 65537 (0x10001)
```

01 02 03 04 05 06 07 08 09 10 11 12 13

How long, from the issued date, is the Kube-API Server Certificate valid for?

File: `/etc/kubernetes/pki/apiserver.crt`

2 Years

6 months

1 Year

10 years

```
  - hostPath:
      path: /var/lib/etcd
      type: DirectoryOrCreate
    name: etcd-data
status: {}
root@controlplane:~# openssl x509 -in /etc/kubernetes/pki/apiserver.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 231788572510502865 (0x3377a7a877bbfd1)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = kubernetes
        Validity
            Not Before: Apr 17 21:17:20 2022 GMT
            Not After : Apr 17 21:17:20 2023 GMT
        Subject: CN = kube-apiserver
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b4:e6:15:51:7c:c9:9e:78:5d:3e:09:85:5b:bf:
                    03:78:d5:4a:44:e7:54:b4:e0:57:7a:28:7b:37:1b:
                    93:49:a7:91:7a:70:3b:bc:fa:28:c6:e4:38:d8:17:
                    b9:98:f7:91:70:e4:64:d8:8a:09:4c:29:00:17:1f:
                    70:01:ef:64:17:25:91:8c:1f:95:aa:f1:68:84:2f:
                    7a:96:45:dc:e8:e8:c2:f7:73:fa:6e:41:b4:ef:89:
                    cd:7e:25:bb:79:60:86:cb:96:96:f3:c7:d7:16:a4:
                    b5:ba:af:27:b8:3e:51:55:11:c7:97:48:c5:f4:89:
                    45:28:d9:0b:b5:f5:fd:13:74:1d:fe:48:02:54:ad:
                    5c:cf:cb:fa:45:cb:5e:47:63:10:7f:7d:45:85:4f:
                    ac:36:50:80:ab:bf:08:cb:cc:ea:53:97:8d:ca:79:
                    3a:67:ed:f7:d7:b4:c7:68:83:34:64:d1:e0:5b:6c:
                    3d:80:c1:5c:5f:a4:c2:ab:03:7f:be:8a:84:7d:30:
                    71:61:bd:37:86:94:99:25:4c:b1:fc:f4:60:56:36:
                    60:d5:d2:be:dd:74:8a:39:6d:84:35:47:5c:08:31:
                    5f:df:ae:e6:c4:08:4a:8c:5d:c2:e1:db:13:45:44:
                    79:cd:55:21:21:1e:20:bf:15:50:f5:b0:4c:97:ab:
                    ae:bd
                Exponent: 65537 (0x10001)
```

01 02 03 04 05 06 07 08 09 10 11 12 13

How long, from the issued date, is the Root CA Certificate valid for?

File: `/etc/kubernetes/pki/ca.crt`

10 years

6 Months

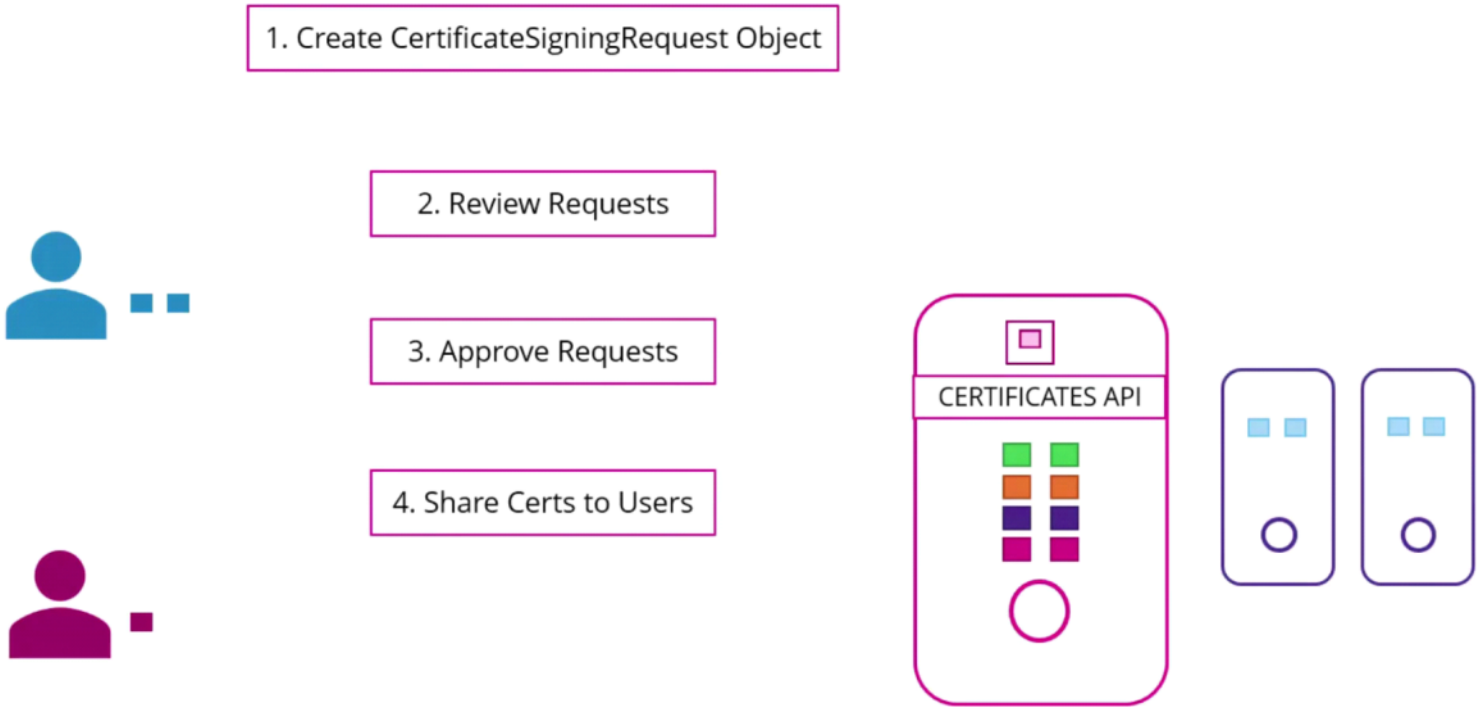
1 Year

2 Years

```
        4e:50:e9:f4:09:7e:05:a5:ed:b0:bb:93:7a:69:0d:66:8e:e3:
        54:80:b0:0c:c5:60:9c:5b:fd:74:6b:bd:4b:c6:40:8e:ee:b2:
        5e:d7:df:5c:c0:52:6b:92:2b:ae:5a:54:c5:5d:e8:01:15:c9:
        cb:15:96:66:58:f6:f7:40:78:1c:8f:9a:a9:8a:69:04:53:f4:
        e9:5e:71:86:cb:2b:7c:e6:aa:39:e2:c5:9d:ad:80:10:f3:a1:
        c1:c8:5f:0c:b2:b6:57:8f:75:db:47:6d:b0:a4:2f:fe:3a:ac:
        55:67:3a:82:0c:22:af:33:2c:10:ff:b2:a8:c2:e0:cd:7a:8e:
        e6:37:88:24:a6:9b:fa:e5:cd:78:48:e6:06:8c:94:63:0d:81:
        21:8b:58:fa:b4:a5:3e:37:a2:d7:0e:f7:fe:4b:87:3d:01:ba:
        70:13:b8:0e:f5:bc:d4:b7:95:8e:fc:60:47:4b:5a:79:f7:a8:
        f1:91:af:e8
root@controlplane:~# openssl x509 -in /etc/kubernetes/pki/ca.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 0 (0x0)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = kubernetes
        Validity
            Not Before: Apr 17 21:17:20 2022 GMT
            Not After : Apr 14 21:17:20 2032 GMT
        Subject: CN = kubernetes
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:a8:16:15:db:54:9f:13:fa:fb:ff:09:c9:a6:6e:
                    b9:ed:2c:c5:fc:2f:6f:56:0c:82:8b:82:32:53:68:
                    67:a2:c7:18:01:48:f2:6e:71:cb:07:7d:27:a4:23:
                    56:22:3f:32:07:80:9b:6f:88:fb:9f:f0:62:89:23:
                    3a:e2:96:95:de:49:f5:03:90:76:4d:61:6d:a4:da:
                    a8:29:5e:c3:62:07:9d:4d:cb:69:ed:c9:88:75:9e:
                    df:96:f5:9c:28:39:45:09:ac:5b:de:34:71:4a:f8:
                    b3:65:11:9e:78:81:1b:4e:99:9c:af:9b:64:1f:03:
                    21:ec:d5:61:66:8b:0c:24:bb:9f:97:b2:30:d0:c3:
                    06:93:27:1d:d9:c1:ef:5e:ee:e3:22:81:48:af:bb:
                    77:5c:fc:fc:b2:8e:e2:18:e4:51:1d:fb:61:0b:e1:
                    69:4c:ce:89:d4:62:79:bd:18:5b:2b:be:ac:89:ba:
                    70:0d:4e:3a:b0:54:e6:bb:b8:d8:a5:5f:97:5e:b0:
```

Certificates API

Friday, June 28, 2024 9:43 PM




```
openssl genrsa -out jane.key 2048
jane.key
```

```
openssl req -new -key jane.key -subj  "/CN=jane" -out jane.csr
```

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

```
kubectl certificate approve jane
jane approved!
```

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

```
echo "LS0…Qo=" | base64 --decode
-----BEGIN CERTIFICATE -----
MIICWDCCAUACAQAwEzERMA8GA1UEAwwIbmV3LXVzZXIwgg
AQUAA4IBDwAwggEKAoIBAQDO0WJW+DXsAJSIrjpNo5vRIE
LfC27t+1eEnON5Muq99NevmMEOnrDUO/thyVqP2w2XNIDR
y3BihhB93MJ7Oql3UTvZ8TELqyaDknRl/jv/SxgXkok0AE
IF5nxAttMVkDPQ7NbeZRG43b+QWlVGR/z6DWOfJnbfezOt
EcCXAwqChjBLkz2BHPR4J89D6Xb8k39pu6jpyngV6uP0tI
j2qEL+hZEWkkFz80lNNtyT5LxMqENDCnIgwC4GZiRGbrAg
9w0BAQsFAAOCAQEAS9iS6C1uxTuf5BBYSU7QFQHUzalNxA
hOK4a2zyNyi44OOijyaD6tUW8DSxkr8BLK8Kg3srREtJqI
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF53lesNSNMLQ2++R
Wr2EUM6UawzykrdHImwTv2mlMY0R+DNtV1Yie+0H9/YElt
4l3E/y3qL71WfAcuH3OsVpUUnQISMdQs0qWCsbE56CC5Dh
vwQ07jG+hpknxmuFAeXxgUwodALaJ7ju/TDIcw==
-----END CERTIFICATE -----
```

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

Practice- cert api

Friday, June 28, 2024 9:49 PM

Task 59:09

01 02 03 04 05 06 07 08 09

A new member `akshay` joined our team. He requires access to our cluster. The Certificate Signing Request is at the `/root` location.

Inspect it

Ok

Terminal 1

```
root@controlplane:~# pwd
/root
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# ls
akshay.csr  akshay.key
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# cat akshay.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICVjCCAT4CAQAwETEPMA0GA1UEAwwGYWtzaGF5MIIBIjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEApzFRPPSJHgmyUQ9jVrdbR9KDN5iFpNRAPGl0T7VhIJ0+
DlvjUApdmnwm++4BLnsL2CLDJb2o25hjMeG4DRkXzScqgLRGoDdEgpEdRRmsENc+
XNmhR03cFJ/SszggMKb0hmFJkHAEFi2WdElBVdXh/lwePhKQwQ2OYmEsALsOODvB
SPVQZnpQRHmTHGjbX9j334TtELkKi79F0Csni2nikqYxICLKP2bKasSf3+9WptBx
86nxl5oGhdgCRqp59E/A/zUOMZ1uTRHhr5w1uRXyivdtiB0a5D34qEpj8shg82HG
uBA7Dmm/ubI7gunXAbY9MKmDIKTxDM0gkT5JMh3/MQIDAQABoAAwDQYJKoZIhvcN
AQELBQADggEBAKM8lqCe7uw7OKMKIPakMtMnUPR3rEff9akKbHhAvNHMCH7JGuoD
7xbo34GELg/v5lsM4xuHCSVIQlkyuixLnrKaQRIV/imPfwzzm2OF6/zmg4tRQRUR
Y1Rz373Dc1NgRMZbGiqoDnEV/gg/F+W0g9lX10UbtP/RQlHGwsn+ZA8/ZPJJxx8n
sFn9vBeeTX9Le8ISHB/WUUlJtnb+gsYFXhMOWsbDkSX99lliD0S6Or4rS2KAPAQO
Eo8UyKr0OFIvCF/s5nk5WwswOlsvGQKrEVKBWwy3Y5O04ERPilC4ImP2yFUMgk3G
yrJeMJpSeRS8d4mJqp3Qkzu1D7QdwJYE23s=
-----END CERTIFICATE REQUEST-----
root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

Task Hint Solution 58:01

01 02 03 04 05 06 07 08 09

Create a CertificateSigningRequest object with the name `akshay` with the contents of the `akshay.csr` file

Check

- CSR akshay created
- Right CSR is used

Terminal 1

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# cat > akshay.yaml
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: myuser
spec:
  request: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJQ1ZqQ0NBVDRDQVFBd0VURVBNQTBHQTFVRUF3d0dZV
zVuWld4aE1JSUJJakFOQmdrcWhraUc5dzBCQVFFRgpBQU9DQVE4QU1JSUJDZ0tDQVFFQTByczhJTHRHdTYxakx2dHhWTTJSVlRWMDNHW
lJTWWw0dWluVWo4RElaWjBOCnR2MUZtRVFSd3VoaUZsOFEzcWl0Qm0wMUFSMkNJVXBGd2ZzSjZ4MXF3ckJzVkhZbGlBNVhwRVpZM3Exc
GswSDQKM3Z3aGJlK1o2MVNrVHF5SVBYUUwrTWM5T1Nsbm0xb0R2N0NtSkZNMUlMRVI3QTVGZnZKOEdFRjJ6dHBoaUlFMwpub1dtdHNZb
3JuT2wzc2lHQ2ZGZzR4Zmd4eW8bymlneFNVekl1bXNnVm9PM2ttT0x1RVF6cXpkakJ3TFJXbWlECklmMXBMWnoyalVnald4UkhCM1gyW
nVVV1d1T09PZnpXM01LaE8ybHEvZi9DdS8wYk83c0x0MCt3U2ZMSU91TFcKcW90blZtRmxMMytqTy82WDNDKzBERHk5aUtwbXJjVDBnW
GZLemE1dHJRSURBUUFCb0FBd0RRWUpLb1pJaHZjTgpBUUVMQlFBRGdnRUJBR05WdmVIOGR4ZzNvK21VeVRkbmFjVmQ1N24zSkExdnZEU
1JWREkyQTZ1eXN3ZFp1L1BVCkkwZXpZWFV0RVNnSk1IRmQycVVNMjNuNVJsSXJ3R0xuUXFISUh5VStWWHhsdnZsRnpNOVpEWllSTmU3Q
lJvYXgKQVlEdUI5STZXT3FYbkFvczFqRmxNUG5NbFpqdU5kSGxpT1BjTU1oNndLaTZzZFhpVStHYTJ2RUVLY01jSVUyRgpvU2djUWdMY
Tk0aEpacGk3ZnNMdm1OQUxoT045UHdNMGM1dVJVejV4T0dGMUtCbWRSeEgvbUNOS2JKYjFRQm1HCkkwYitEUEdaTktXTU0xMzhIQXdoV
0tkNjVoVHdYOWl4V3ZHMkh4TG1WQzg0L1BHT0tWQW9FNkpsYWFHdTlQVmkKdjlOSjVaZlZrcXdCd0hKbzZXdk9xVlA3SVFjZmg3d0drW
m89Ci0tLS0tRU5EIENFUlRJRklDQVRFIFJFUVVFU1QtLS0tLQo=
  signerName: kubernetes.io/kube-apiserver-client
  expirationSeconds: 86400  # one day
  usages:
  - client auth

^C
root@controlplane:~# cat akshay.
```

Task Hint Solution 57:58

01 02 03 04 05 06 07 08 09

Create a CertificateSigningRequest object with the name `akshay` with the contents of the `akshay.csr` file

Check

- CSR akshay created
- Right CSR is used

Terminal 1 +

```
kind: CertificateSigningRequest
metadata:
  name: myuser
spec:
  request: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJQ1ZqQ0NBVDRDQVFBd0VURVBNQTBHQTFVRUF3d0dZV
zVuWld4aE1JSUJJakFOQmdrcWhraUc5dzBCQVFFRgpBQU9DQVE4QU1JSUJDZ0tDQVFFQTByczhJTHRHdTYxakx2dHhWTTJSVlRWMDNHW
lJTWWw0dWluVWo4RElaWjBOCnR2MUZtRVFSd3VoaUZsOFEzcWl0Qm0wMUFSMkNJVXBGd2ZzSjZ4MXF3ckJzVkhZbGlBNVhwRVpZM3Exc
GswSDQKM3Z3aGJlK1o2MVNrVHF5SVBYUUwrTWM5T1Nsbm0xb0R2N0NtSkZNMUlMRVI3QTVGZnZKOEdFRjJ6dHBoaUlFMwpub1dtdHNZb
3JuT2wzc2lHQ2ZGZzR4Zmd4eW8ybmlneFNVekl1bXNnVm9PM2ttT0x1RVF6cXpkakJ3TFJXbWlECklmMXBMWnoyalVnald4UkhCM1gyW
nVVV1d1T09PZnpXM01LaE8ybHEvZi9DdS8wYk83c0x0MCt3U2ZMSU91TFcKcW90blZtRmxMMytqTy8ZWDNDKzBERHk5aUtwbXJjVDBnW
GZLemE1dHJRSURBUUFCb0FBd0RRWUpLb1pJaHZjTgpBUUVMQlFBRGdnRUJBR05WdmVIOGR4ZzNvK21VeVRkbmFjVmQ1N24zSkExdnZEU
1JWREkyQTZ1eXN3ZFp1L1BVCkkwZXpZWFV0RVNnSk1IRmQycVVNMjNuNVJsSXJ3R0xuUXFISUh5VStWWHhsdnZsRnpNOVpEWllSTmU3Q
lJvYXgKQVlEdUI5STZXT3FYbkFvczFqRmxNUG5NbFpqdU5kSGxpT1BjTU1oNndLaTZzZFhpVStHYTJ2RUVLY01jSVUyRgpvU2djUWdMY
Tk0aEpacGk3ZnNMdm1OQUxoT045UHdNMGM1dVJVejV4T0dGMUtCbWRSeEgvbUNOS2JKYjFRQm1HCkkwYitEUEdaTktXTU0xMzhIQXdoV
0tkNjVoVHdYOWl4V3ZHMkh4TG1WQzg0L1BHT0tWQW9FNkpsYWFHdTlQVmkKdjlOSjVaZlZrcXdCd0hKbzZXdk9xVlA3SVFjZmg3d0drW
m89Ci0tLS0tRU5EIENFUlRJRklDQVRFIFJFUVVFU1QtLS0tLQo=
  signerName: kubernetes.io/kube-apiserver-client
  expirationSeconds: 86400  # one day
  usages:
  - client auth

^C
root@controlplane:~# cat akshay.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICVjCCAT4CAQAwETEPMA0GA1UEAwwGYWtzaGF5MIIBIjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEApzFRPPSJHgmyUQ9jVrdbR9KDN5iFpNRAPGl0T7VhIJ0+
DlvjUApdmnwm++4BLnsL2CLDJb2o25hjMeG4DRkXzScqgLRGoDdEgpEdRRmsENc+
XNmhR03cFJ/SszggMKbOhmFJkHAEFi2WdElBVdXh/lwePhKQwQ2OYmEsALsOODvB
SPVQZnpQRHmTHGjbX9j334TtELkKi79F0Csni2nikqYxICLKP2bKasSf3+9WptBx
86nxl5oGhdgCRqp59E/A/zUOMZ1uTRHhr5w1uRXyivdtiB0a5D34qEpj8shg82HG
uBA7Dmm/ubI7gunXAbY9MKmDIKTxDM0gkT5JMh3/MQIDAQABoAAwDQYJKoZIhvcN
AQELBQADggEBAKM8lqCe7uw7OKMKIPakMtMnUPR3rEff9akKbHhAvNHMCH7JGuoD
7xbo34GELg/v5lsM4xuHCSVIQlkyuixLnrKaQRIV/imPfwzzm2OF6/zmg4tRQRUR
Y1Rz373Dc1NgRMZbGiqoDnEV/gg/F+W0g9lX10UbtP/RQlHGwsn+ZA8/ZPJJxx8n
sFn9vBeeTX9Le8ISHB/WUUlJtnb+gsYFXhMOWsbDkSX99lliD0S6Or4rS2KAPAQO
Eo8UyKr0OFIvCF/s5nk5WwswOlsvGQKrEVKBWwy3Y5O04ERPilC4ImP2yFUMgk3G
yrJeMJpSeRS8d4mJqp3Qkzu1D7QdwJYE23s=
-----END CERTIFICATE REQUEST-----
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09

Create a CertificateSigningRequest object with the name `akshay` with the contents of the `akshay.csr` file

Check

- CSR akshay created
- Right CSR is used

```
Y1Rz373Dc1NgRMZbGiqoDnEV/gg/F+W0g9lX10UbtP/RQlHGwsn+ZA8/ZPJJxx8n
sFn9vBeeTX9Le8ISHB/WUUlJtnb+gsYFXhMOWsbDkSX99lliD0S6Or4rS2KAPAQO
Eo8UyKr0OFIvCF/s5nk5WwswOlsvGQKrEVKBWwy3Y5O04ERPilC4ImP2yFUMgk3G
yrJeMJpSeRS8d4mJqp3Qkzu1D7QdwJYE23s=
-----END CERTIFICATE REQUEST-----
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# cat akshay.csr | base64
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJQ1ZqQ0NBVDRDQVFBd0VURVBN
QTBHQTFVRUF3d0dZV3R6YUdGNU1JSUJJakFOQmdrcWhraUc5dzBCQVFFRgpBQU9DQVE4QU1JSUJD
Z0tDQVFFQXB6RlJQUFNKSGdteVVROWpWcmRiUjlLRE41aUZwTlJBUEdsMFQ3VmhJSjArCkRsdmpV
QXBkbW53bSsrNEJMbnNMMkNMREpiMm8yNWhqTWVHNERSa1h6U2NxZ0xSR29EZEVncEVkUlJtc0VO
YysKWE5taFIwM2NGSi9Tc3pnZ01LYk9obUZKa0hBRUZpMldkRWxCVmRYaC9sd2VQaEtRd1EyT1lt
RXNBTHNPT0R2QgpTUFZRWm5wUVJIbVRIR2piWDlqMzM0VHRFTGtLaTc5RjBDc25pMm5pa3FZeElD
TEtQMmJLYXNTZjMrOVdwdEJ4Cjg2bnhsNW9HaGRnQ1JxcDU5RS9BL3pVT01aMXVUUkhocjV3MXVS
WHlpdmR0aUIwYTVEMzRxRXBqOHNoZzgySEcKdUJBN0RtbS91Ykk3Z3VuWEFiWTlNS21ESUtUeERN
MGdrVDVKTWgzL01RSURBUUFCb0FBd0RRWUpLb1pJaHZjTgpBUUVMQlFBRGdnRUJBS004bHFDZTd1
dzdPS01LSVBha010TW5VUFIzckVmZjlha0tiSGhBdk5ITUNIN0pHdW9ECjd4Ym8zNEdFTGcvdjVs
c000eHVIQ1NWSVFsa3l1aXhMbnJLYVFSSVYvaW1QZnd6em0yT0Y2L3ptZzR0UlFSVVIKWTFSejM3
M0RjMU5nUk1aYkdpcW9EbkVWL2dnL0YrVzBnOWxYMTBVYnRQL1JRbEhHd3NuK1pBOC9aUEpKeHg4
bgpzRm45dkJlZVRYOUxlOElTSEIvV1VVbEp0bmIrZ3NZRlhoTU9Xc2JEa1NYOTlsbGlEMFM2T3I0
clMyS0FQQVFPCkVvOFV5S3IwT0ZJdkNGL3M1bms1V3dzd09sc3ZHUUtyRVZLQld3eTNZNU8wNEVS
UGlsQzRJbVAyeUZVTWdrM0cKeXJKZU1KcFNlUlM4ZDRtSnFwM1FrenUxRDdRZHdKWUUyM3M9Ci0t
LS0tRU5EIENFUlRJRklDQVRFIFJFUVVFU1QtLS0tLQo=
root@controlplane:~# cat akshay.csr | base64 -w 0
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJQ1ZqQ0NBVDRDQVFBd0VURVBNQTBHQTFVRUF3d0dZV3R6YUdGNU1J
SUJJakFOQmdrcWhraUc5dzBCQVFFRgpBQU9DQVE4QU1JSUJDZ0tDQVFFQXB6RlJQUFNKSGdteVVROWpWcmRiUjlLRE41aUZwTlJBUEds
MFQ3VmhJSjArCkRsdmpVQXBkbW53bSsrNEJMbnNMMkNMREpiMm8yNWhqTWVHNERSa1h6U2NxZ0xSR29EZEVncEVkUlJtc0VOYysKWE5t
aFIwM2NGSi9Tc3pnZ01LYk9obUZKa0hBRUZpMldkRWxCVmRYaC9sd2VQaEtRd1EyT1ltRXNBTHNPT0R2QgpTUFZRWm5wUVJIbVRIR2pi
WDlqMzM0VHRFTGtLaTc5RjBDc25pMm5pa3FZeElDTEtQMmJLYXNTZjMrOVdwdEJ4Cjg2bnhsNW9HaGRnQ1JxcDU5RS9BL3pVT01aMXVU
UkhocjV3MXVSWHlpdmR0aUIwYTVEMzRxRXBqOHNoZzgySEcKdUJBN0RtbS91Ykk3Z3VuWEFiWTlNS21ESUtUeERNMGdrVDVKTWgzL01R
SURBUUFCb0FBd0RRWUpLb1pJaHZjTgpBUUVMQlFBRGdnRUJBS004bHFDZTd1dzdPS01LSVBha010TW5VUFIzckVmZjlha0tiSGhBdk5I
TUNIN0pHdW9ECjd4Ym8zNEdFTGcvdjVsc000eHVIQ1NWSVFsa3l1aXhMbnJLYVFSSVYvaW1QZnd6em0yT0Y2L3ptZzR0UlFSVVIKWTFS
ejM3M0RjMU5nUk1aYkdpcW9EbkVWL2dnL0YrVzBnOWxYMTBVYnRQL1JRbEhHd3NuK1pBOC9aUEpKeHg4bgpzRm45dkJlZVRYOUxlOElT
SEIvV1VVbEp0bmIrZ3NZRlhoTU9Xc2JEa1NYOTlsbGlEMFM2T3I0clMyS0FQQVFPCkVvOFV5S3IwT0ZJdkNGL3M1bms1V3dzd09sc3ZH
UUtyRVZLQld3eTNZNU8wNEVSUGlsQzRJbVAyeUZVTWdrM0cKeXJKZU1KcFNlUlM4ZDRtSnFwM1FrenUxRDdRZHdKWUUyM3M9Ci0tLS0t
RU5EIENFUlRJRklDQVRFIFJFUVVFU1QtLS0tLQo=root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

```
root@controlplane:~#
root@controlplane:~# vi akshay.csr
root@controlplane:~# vi akshay.yaml
root@controlplane:~# vi akshay.yaml
root@controlplane:~# kubectl create -f akshay.yaml
Error from server (BadRequest): error when creating "akshay.yaml": CertificateSigningRequest in version
"v1" cannot be handled as a CertificateSigningRequest: v1.CertificateSigningRequest.Spec: v1.Certificate
SigningRequestSpec.Usages: []v1.KeyUsage: SignerName: Request: decode base64: illegal base64 data at inp
ut byte 1180, error found in #10 byte of ...|QtLS0tLQo","signerNa|..., bigger context ...|M9Ci0tLS0tRU5E
IENFUlRJRklDQVRFIFJFUVVFU1QtLS0tLQo","signerName":"kubernetes.io/kube-apiserver-client|...
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# vi akshay.yaml
root@controlplane:~# vi akshay.yaml
root@controlplane:~#
root@controlplane:~# kubectl create -f akshay.yaml
certificatesigningrequest.certificates.k8s.io/akshay created
root@controlplane:~#
root@controlplane:~# kubectl get csr
NAME        AGE   SIGNERNAME                                    REQUESTOR                  CONDITION
akshay      4s    kubernetes.io/kube-apiserver-client           kubernetes-admin           Pending
csr-8mfn8   29m   kubernetes.io/kube-apiserver-client-kubelet   system:node:controlplane   Approved,Issu
ed
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09

What is the Condition of the newly created Certificate Signing Request object?

Approved

Issued

Pending

```
root@controlplane:~# cat akshay.csr | base64 -w 0
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJQ1ZqQ0NBVDRDQVFBd0VURVBNQTBHQTFVRUF3d0dZV3R6YUdGNU1J
SUJJakFOQmdrcWhraUc5dzBCQVFFRgpBQU9DQVE4QU1JSUJDZ0tDQVFFQXB6RlJQUFNKSGdteVVROWpWcmRiUjlLRE41aUZwTlJBUEds
MFQ3VmhJSjArCkRsdmpVQXBkbW53bSsrNEJMbnNMMkNMREpiMm8yNWhqTWVHNERSa1h6U2NxZ0xSR29EZEVncEVkUlJtc0VOYysKWE5t
aFIwM2NGSi9Tc3pnZ01LYk9obUZKa0hBRUZpMldkRWxCVmRYaC9sd2VQaEtRd1EyT1ltRXNBTHNPT0R2QgpTUFZRWm5wUVJIbVRIR2pi
WDlqMzM0VHRFTGtLaTc5RjBDc25pMm5pa3FZeElDTEtQMmJLYXNTZjMrOVdwdEJ4Cjg2bnhsNW9HaGRnQ1JxcDU5RS9BL3pVT01aMXVU
UkhocjV3MXVSWHlpdmR0aUIwYTVEMzRxRXBqOHNoZzgySEcKdUJBN0RtbS91Ykk3Z3VuWEFiWTlNS21ESUtUeERNMGdrVDVKTWgzL01R
SURBUUFCb0FBd0RRWUpLb1pJaHZjTgpBUUVMQlFBRGdnRUJBS004bHFDZTd1dzdPS01LSVBha010TW5VUFIzckVmZjlha0tiSGhBdk5I
TUNIN0pHdW9ECjd4Ym8zNEdFTGcvdjVsc000eHVIQ1NWSVFsa3l1aXhMbnJLYVFSSVYvaW1QZnd6em0yT0Y2L3ptZzR0UlFSVVIKWTFS
ejM3M0RjMU5nUk1aYkdpcW9EbkVWL2dnL0YrVzBnOWxYMTBVYnRQL1JRbEhHd3NuK1pBOC9aUEpKeHg4bgpzRm45dkJlZVRYOUxlOElT
SEIvV1VVbEp0bmIrZ3NZRlhoTU9Xc2JEa1NYOTlsbGlEMFM2T3I0clMyS0FQQVFPCkVvOFV5S3IwT0ZJdkNGL3M1bms1V3dzd09sc3ZH
UUtyRVZLQld3eTNZNU8wNEVSUGlsQzRJbVAyeUZVTWdrM0cKeXJKZU1KcFNlUlM4ZDRtSnFwM1FrenUxRDdRZHdKWUUyM3M9Ci0tLS0t
RU5EIENFUlRJRklDQVRFIFJFUVVFU1QtLS0tLQo=root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# vi akshay.csr
root@controlplane:~# vi akshay.yaml
root@controlplane:~# vi akshay.yaml
root@controlplane:~# kubectl create -f akshay.yaml
Error from server (BadRequest): error when creating "akshay.yaml": CertificateSigningRequest in version
"v1" cannot be handled as a CertificateSigningRequest: v1.CertificateSigningRequest.Spec: v1.Certificate
SigningRequestSpec.Usages: []v1.KeyUsage: SignerName: Request: decode base64: illegal base64 data at inp
ut byte 1180, error found in #10 byte of ...|QtLS0tLQo","signerNa|..., bigger context ...|M9Ci0tLS0tRU5E
IENFUlRJRklDQVRFIFJFUVVFU1QtLS0tLQo","signerName":"kubernetes.io/kube-apiserver-client|...
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# vi akshay.yaml
root@controlplane:~# vi akshay.yaml
root@controlplane:~#
root@controlplane:~# kubectl create -f akshay.yaml
certificatesigningrequest.certificates.k8s.io/akshay created
root@controlplane:~#
root@controlplane:~# kubectl get csr
NAME        AGE   SIGNERNAME                                    REQUESTOR                  CONDITION
akshay      4s    kubernetes.io/kube-apiserver-client           kubernetes-admin           Pending
csr-8mfn8   29m   kubernetes.io/kube-apiserver-client-kubelet   system:node:controlplane   Approved,Issu
ed
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09

How many CSR requests are available on the cluster?

Including approved and pending

```
root@controlplane:~#
root@controlplane:~# kubectl certificate approve akshay
certificatesigningrequest.certificates.k8s.io/akshay approved
```

```
csr-8mfn8   29m   kubernetes.io/kube-apiserver-client-kubelet   system:node:controlplane   Approved,Issu
ed
root@controlplane:~#
root@controlplane:~# kubectl certificate approve akshay
certificatesigningrequest.certificates.k8s.io/akshay approved
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get csr
NAME        AGE   SIGNERNAME                                    REQUESTOR                 CONDITION
akshay      42s   kubernetes.io/kube-apiserver-client           kubernetes-admin          Approved,Issu
ed
csr-8mfn8   30m   kubernetes.io/kube-apiserver-client-kubelet   system:node:controlplane   Approved,Issu
ed
root@controlplane:~#
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09

Approve the CSR Request

Next

```
RU5EIENFUlRJRklDQVRFIFJFUVVFU1QtLS0tLQo=root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# vi akshay.csr
root@controlplane:~# vi akshay.yaml
root@controlplane:~# vi akshay.yaml
root@controlplane:~# kubectl create -f akshay.yaml
Error from server (BadRequest): error when creating "akshay.yaml": CertificateSigningRequest in version
"v1" cannot be handled as a CertificateSigningRequest: v1.CertificateSigningRequest.Spec: v1.Certificate
SigningRequestSpec.Usages: []v1.KeyUsage: SignerName: Request: decode base64: illegal base64 data at inp
ut byte 1180, error found in #10 byte of ...|QtLS0tLQo","signerNa|..., bigger context ...|M9Ci0tLS0tRU5E
IENFUlRJRklDQVRFIFJFUVVFU1QtLS0tLQo","signerName":"kubernetes.io/kube-apiserver-client|...
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# vi akshay.yaml
root@controlplane:~# vi akshay.yaml
root@controlplane:~#
root@controlplane:~# kubectl create -f akshay.yaml
certificatesigningrequest.certificates.k8s.io/akshay created
root@controlplane:~#
root@controlplane:~# kubectl get csr
NAME        AGE   SIGNERNAME                                    REQUESTOR                 CONDITION
akshay      4s    kubernetes.io/kube-apiserver-client           kubernetes-admin          Pending
csr-8mfn8   29m   kubernetes.io/kube-apiserver-client-kubelet   system:node:controlplane   Approved,Issu
ed
root@controlplane:~#
root@controlplane:~# kubectl certificate approve akshay
certificatesigningrequest.certificates.k8s.io/akshay approved
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get csr
NAME        AGE   SIGNERNAME                                    REQUESTOR                 CONDITION
akshay      42s   kubernetes.io/kube-apiserver-client           kubernetes-admin          Approved,Issu
ed
csr-8mfn8   30m   kubernetes.io/kube-apiserver-client-kubelet   system:node:controlplane   Approved,Issu
ed
root@controlplane:~#
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09

Hmmm.. You are not aware of a request coming in. What groups is this CSR requesting access to?

Check the details about the request. Preferebly in YAML.

- system:masters
- certificateapprovals
- system:apiserver
- system:nodes

```
Error from server (BadRequest): error when creating "akshay.yaml": CertificateSigningRequest in version
"v1" cannot be handled as a CertificateSigningRequest: v1.CertificateSigningRequest.Spec: v1.Certificate
SigningRequestSpec.Usages: []v1.KeyUsage: SignerName: Request: decode base64: illegal base64 data at inp
ut byte 1180, error found in #10 byte of ...|QtLS0tLQo","signerNa|..., bigger context ...|M9Ci0tLS0tRU5E
IENFUlRJRklDQVRFIFJFUVVFU1QtLS0tLQo","signerName":"kubernetes.io/kube-apiserver-client|...
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# vi akshay.yaml
root@controlplane:~# vi akshay.yaml
root@controlplane:~#
root@controlplane:~# kubectl create -f akshay.yaml
certificatesigningrequest.certificates.k8s.io/akshay created
root@controlplane:~#
root@controlplane:~# kubectl get csr
NAME        AGE   SIGNERNAME                                    REQUESTOR                   CONDITION
akshay      4s    kubernetes.io/kube-apiserver-client           kubernetes-admin            Pending
csr-8mfn8   29m   kubernetes.io/kube-apiserver-client-kubelet   system:node:controlplane    Approved,Issu
ed
root@controlplane:~#
root@controlplane:~# kubectl certificate approve akshay
certificatesigningrequest.certificates.k8s.io/akshay approved
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get csr
NAME        AGE   SIGNERNAME                                    REQUESTOR                   CONDITION
akshay      42s   kubernetes.io/kube-apiserver-client           kubernetes-admin            Approved,Issu
ed
csr-8mfn8   30m   kubernetes.io/kube-apiserver-client-kubelet   system:node:controlplane    Approved,Issu
ed
root@controlplane:~#
root@controlplane:~# kubectl get csr
NAME          AGE   SIGNERNAME                                    REQUESTOR                   CONDITION
agent-smith   7s    kubernetes.io/kube-apiserver-client           agent-x                     Pending
akshay        67s   kubernetes.io/kube-apiserver-client           kubernetes-admin            Approved,Is
sued
csr-8mfn8     30m   kubernetes.io/kube-apiserver-client-kubelet   system:node:controlplane    Approved,Is
sued
root@controlplane:~#
root@controlplane:~# kubectl get csr agent-smith -o yaml
```

```
  managedFields:
  - apiVersion: certificates.k8s.io/v1
    fieldsType: FieldsV1
    fieldsV1:
      f:spec:
        f:groups: {}
        f:request: {}
        f:signerName: {}
        f:usages: {}
    manager: kubectl-create
    operation: Update
    time: "2022-04-17T22:29:56Z"
  name: agent-smith
  resourceVersion: "2663"
  uid: 7a31026a-0e86-4986-8680-13753ba05ed2
spec:
  groups:
  - system:masters
  - system:authenticated
  request: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJQ1dEQ0NBVUFDQVFBd0
VYzTFhWelpYSXdnZ0VpTUEwR0NTcUdTSWIzRFFFQgpBUVVBQTRJQkR3QXdnZ0VLQW9JQkFRRE8wV0pXK0
GxuemcrNnhjOStVVndrS2kwCkxmQzI3dCsxZUVuT041TXVxOTlOZXZtTUVPbnJEVU8vdGh5VnFQMncyWE
ld5Q0sKeTNCaWhoQjkzTUo3T3FsM1VUdlo4VEVMcXlhRGtuUmwvanYvU3hnWGtvazBBQlVUcFdNeDRCcF
E1Wa0RQUTdOYmVaUkc0M2IrUVdsVkdSL3o2RFdPZkpuYmZlek90YUF5ZEdMVFpGQy93VHB6NTJrCkVjQ1
jg5RDZYYjhrMzlwdTZqcHluZ1Y2dVAwdEliT3pwcU52MFkwcWRFWnB3bXcKajJxRUwraFpFV2trRno4MG
3dDNEdaaVJHYnJBZ01CQUFHZ0FEQU5CZ2txaGtpRwo5dzBCQVFzRkFBT0NBUUVBUzlpUzZDMXV4VHVmNU
FlzYU9SUlFOd0had0hxR2k0CmhPSzRhMnp5TnlpNDRPT2lqeWFENnRVVzhEU3hrcjhCTEs4S2czc3JSRX
EQ0Z1kKUDlOTCthRFJTeFJPVlNxQmFCMm5XZVlwTTVjSjVURjUzbGVzTlNOTUxRMisrUk1uakRRSjdqdV
WF3enlrcmRISW13VHYybWxNWTBSK0ROdFYxWWllKzBIOS9ZRWx0K0ZTR2poNUw1WVV2STFEcWl5CjRsM0
nBVVW5RSVNNZFFzMHFXQ3NiRTU2Q0M1RGhQR1pJcFVibktVcEF3a2ErOEUKdndRMDdqRytocGtueG11Rk
ERJY3c9PQotLS0tLUVORCBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0K
  signerName: kubernetes.io/kube-apiserver-client
  usages:
  - digital signature
  - key encipherment
  - server auth
  username: agent-x
status: {}
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09

That doesn't look very right. Reject that request.

Next

```
        f:request: {}
        f:signerName: {}
        f:usages: {}
    manager: kubectl-create
    operation: Update
    time: "2022-04-17T22:29:56Z"
  name: agent-smith
  resourceVersion: "2663"
  uid: 7a31026a-0e86-4986-8680-13753ba05ed2
spec:
  groups:
  - system:masters
  - system:authenticated
  request: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJQ1dEQ0NBVUFDQVFBd0V6RVJNQThHQTFVRUF3d0lib
VYzTFhWelpYSXdnZ0VpTUEwR0NTcUdTSWIzRFFFQgpBUVVBQTRJQkR3QXdnZ0VLQW9JQkFRRE8wV0pXK0RYc0FKU0lyanBObzV2UklCc
GxuemcrNnhjOStVVndrS2kwCkxmQzI3dCsxZUVuT041TXVxOTlOZXZtTUVPbnJEVU8vdGh5VnFQMncyWE5JRFJYall5RjQwRmJtRCs1e
ld5Q0sKeTNCaWhoQjkzTUo3T3FsM1VUdlo4VEVMcXlhRGtuUmwvanYvU3hnWGtvazBBQlVUcFdNeDRCcFNpS2IwVSt0RQpJRjVueEF0d
E1Wa0RQUTdOYmVaUkc0M2IrUVdsVkdSL3o2RFdPZkpuYmZlek90YUF5ZEdMVFpGQy93VHB6NTJrCkVjQ1hBd3FDaGpCTGt6MkJIUFI0S
jg5RDZYYjhrMzlwdTZqcHluZ1Y2dVAwdEliT3pwcU52MFkwcWRFWnB3bXcKajJxRUwraFpFV2trRno4MGxOTnR5VDVMeE1xRU5EQ25JZ
3dDNEdaaVJHYnJBZ01CQUFHZ0FEQU5CZ2txaGtpRwo5dzBCQVFzRkFBT0NBUUVBUzlpUzZDMXV4VHVmNUJCWVNVN1FGUUhVemFsTnhBZ
FlzYU9SUlFOd0had0hxR2k0CmhPSzRhMnp5TnlpNDRPT2lqeWFENnRVVzhEU3hrcjhCTEs4S2czc3JSRXRKcWw1ckxaeTlMUlZyc0pna
EQ0Z1kKUDlOTCthRFJTeFJPVlNxQmFCMm5XZVlwTTVjSjVURjUzbGVzTlNOTUxRMisrUk1uakRRSjdqdVBFaWM4L2RoawpXcjJFVU02V
WF3enlrcmRISW13VHYybWxNWTBSK0ROdFYxWWllKzBIOS9ZRWx0K0ZTR2poNUw1WVV2STFEcWl5CjRsM0UveTNxTDcxV2ZBY3VIM09zV
nBVVW5RSVNNZFFzMHFXQ3NiRTU2Q0M1RGhQR1pJcFVibktVcEF3a2ErOEUKdndRMDdqRytocGtueG11RkFlWHhnVXdvZEFMYUo3anUvV
ERJY3c9PQotLS0tLUVORCBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0K
  signerName: kubernetes.io/kube-apiserver-client
  usages:
  - digital signature
  - key encipherment
  - server auth
  username: agent-x
status: {}
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl certificate deny agent-smith
certificatesigningrequest.certificates.k8s.io/agent-smith denied
root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09

Let's get rid of it. Delete the new CSR object

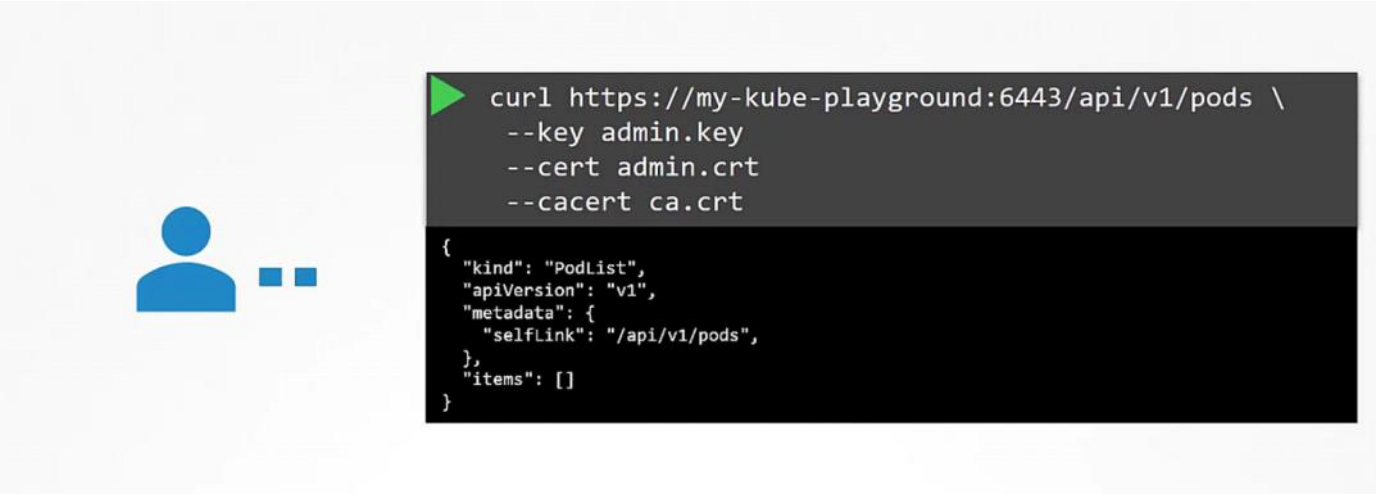
Complete

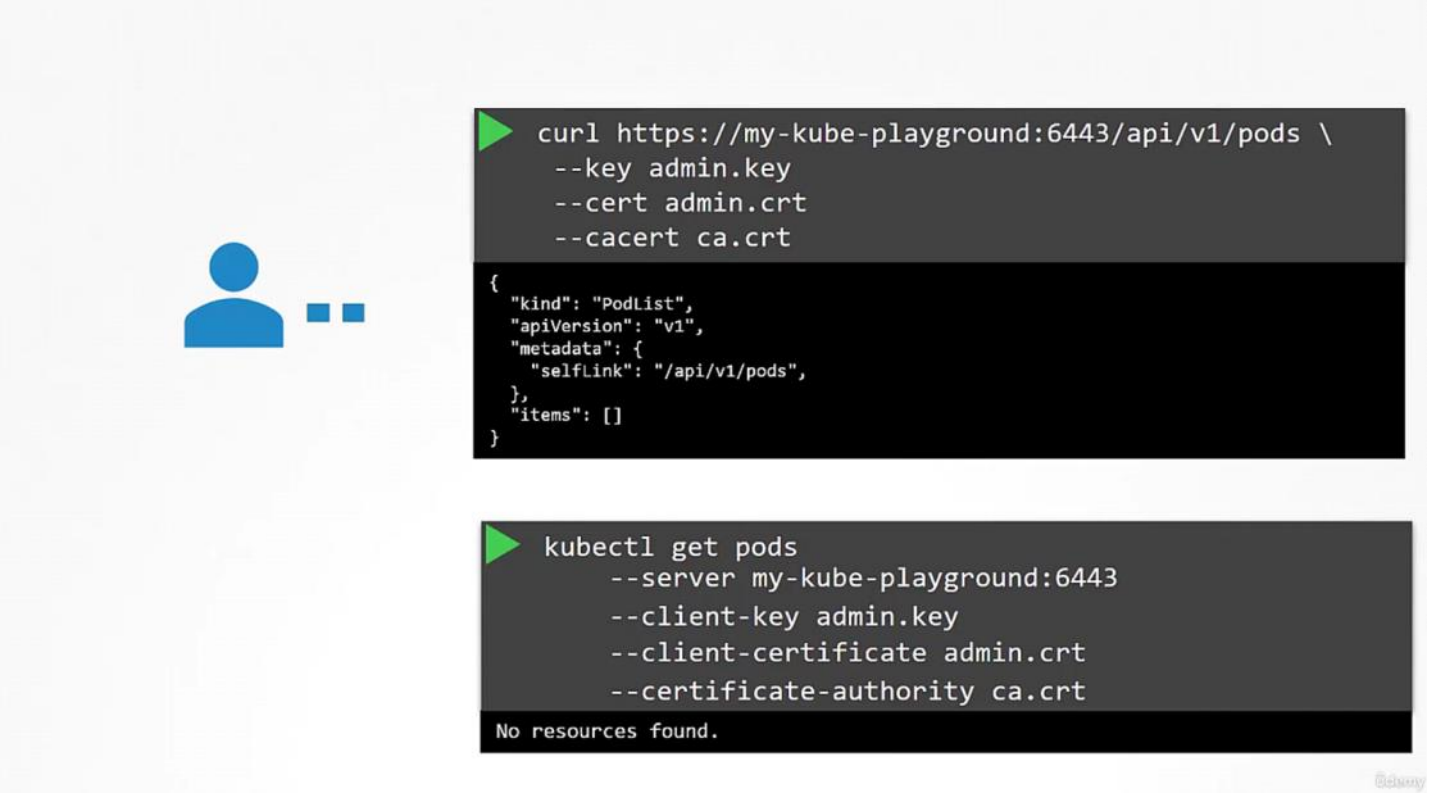
Next

```
3dDNEdaaVJHYnJBZ01CQUFHZ0FEQU5CZ2txaGtpRwo5dzBCQVFzRkFBT0NBUUVBUzlpUzZDMXV4VHVmNUJCWVNVN1FGUUhVemFsTnhBZ
FlzYU9SUlFOd0had0hxR2k0CmhPSzRhMnp5TnlpNDRPT2lqeWFENnRVVzhEU3hrcjhCTEs4S2czc3JSRXRKcWw1ckxaeTlMUlZyc0pna
EQ0Z1kKUDlOTCthRFJTeFJPVlNxQmFCMm5XZVlwTTVjSjVURjUzbGVzTlNOTUxRMisrUk1uakRRSjdqdVBFaWM4L2RoawpXcjJFVU02V
WF3enlrcmRISW13VHYybWxNWTBSK0ROdFYxWWllKzBIOS9ZRWx0K0ZTR2poNUw1WVV2STFEcWl5CjRsM0UveTNxTDcxV2ZBY3VIM09zV
nBVVW5RSVNNZFFzMHFXQ3NiRTU2Q0M1RGhQR1pJcFVibktVcEF3a2ErOEUKdndRMDdqRytocGtueG11RkFlWHhnVXdvZEFMYUo3anUvV
ERJY3c9PQotLS0tLUVORCBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0K
  signerName: kubernetes.io/kube-apiserver-client
  usages:
  - digital signature
  - key encipherment
  - server auth
  username: agent-x
status: {}
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl certificate deny agent-smith
certificatesigningrequest.certificates.k8s.io/agent-smith denied
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get csr
NAME          AGE     SIGNERNAME                                    REQUESTOR                  CONDITION
agent-smith   103s    kubernetes.io/kube-apiserver-client           agent-x                    Denied
akshay        2m43s   kubernetes.io/kube-apiserver-client           kubernetes-admin           Approved,
Issued
csr-8mfn8     32m     kubernetes.io/kube-apiserver-client-kubelet   system:node:controlplane   Approved,
Issued
root@controlplane:~# kubectl delete csr agent-smith
certificatesigningrequest.certificates.k8s.io "agent-smith" deleted
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get csr
NAME        AGE     SIGNERNAME                                    REQUESTOR                  CONDITION
akshay      2m55s   kubernetes.io/kube-apiserver-client           kubernetes-admin           Approved,Is
sued
csr-8mfn8   32m     kubernetes.io/kube-apiserver-client-kubelet   system:node:controlplane   Approved,Is
sued
root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

Kube Config

Saturday, June 29, 2024 9:21 AM





$HOME/.kube/config

KubeConfig File

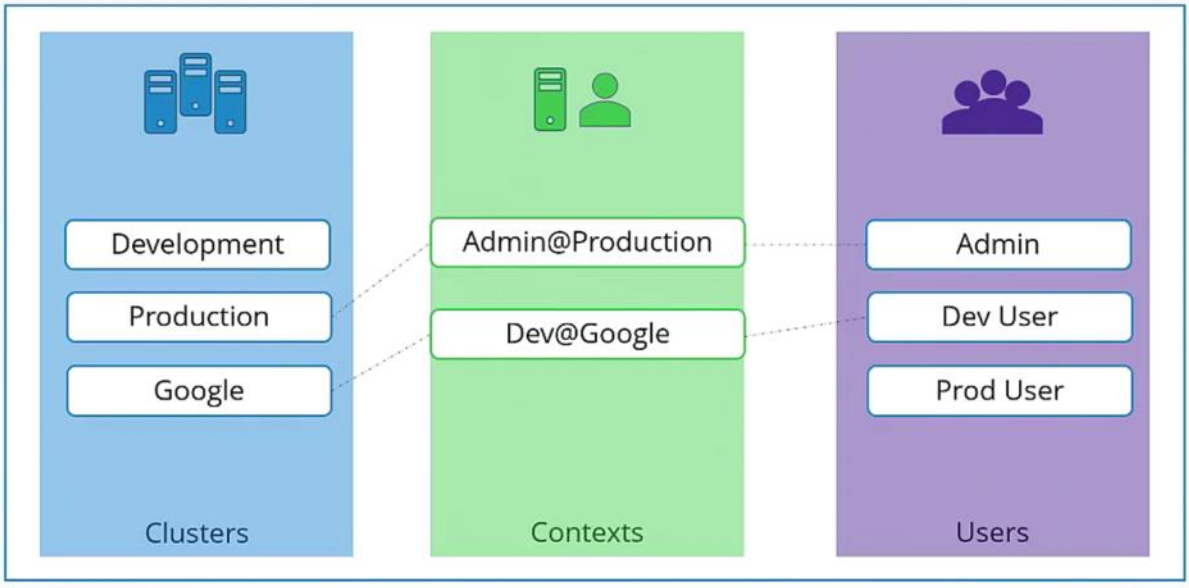
```
--server my-kube-playground:6443
--client-key admin.key
--client-certificate admin.crt
--certificate-authority ca.crt
```

```
kubectl get pods
```

```
No resources found.
```

KubeConfig File

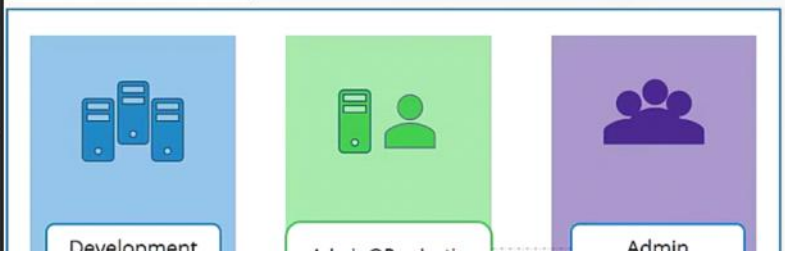
$HOME/.kube/config



KubeConfig File

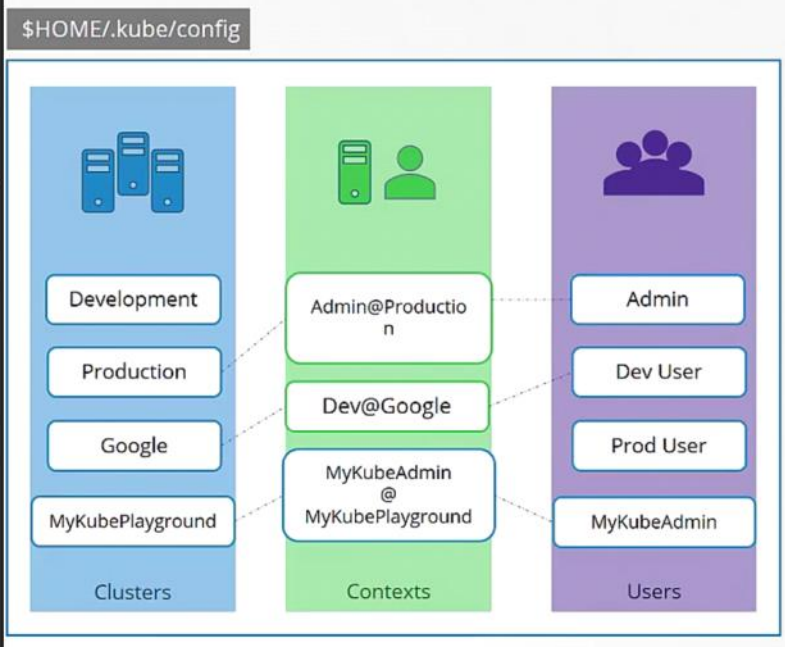
```yaml
apiVersion: v1
kind: Config

clusters:
- name: my-kube-playground
  cluster:
    certificate-authority: ca.crt
    server: https://my-kube-playground:6443

contexts:
```

$HOME/.kube/config



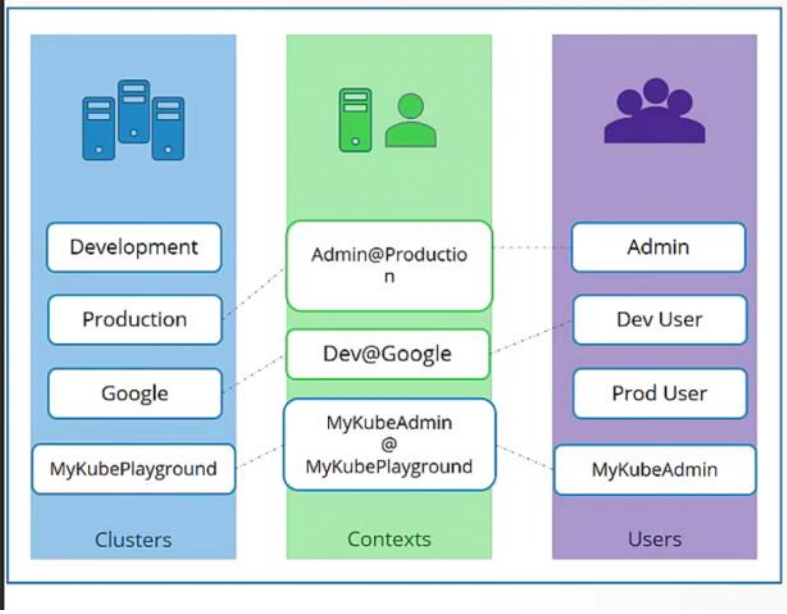
KubeConfig File

```yaml
apiVersion: v1
kind: Config

clusters:
- name: my-kube-playground
  cluster:
    certificate-authority: ca.crt
    server: https://my-kube-playground:6443

contexts:
- name: my-kube-admin@my-kube-playground
  context:
    cluster: my-kube-playground
    user: my-kube-admin

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



KubeConfig File

```yaml
apiVersion: v1
kind: Config

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
```



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
```

Kubectl config

```
kubectl config -h
Available Commands:
  current-context Displays the current-context
  delete-cluster  Delete the specified cluster from the kubeconfig
  delete-context  Delete the specified context from the kubeconfig
  get-clusters    Display clusters defined in the kubeconfig
  get-contexts    Describe one or many contexts
  rename-context  Renames a context from the kubeconfig file.
  set             Sets an individual value in a kubeconfig file
  set-cluster     Sets a cluster entry in kubeconfig
  set-context     Sets a context entry in kubeconfig
  set-credentials Sets a user entry in kubeconfig
  unset           Unsets an individual value in a kubeconfig file
  use-context     Sets the current-context in a kubeconfig file
  view            Display merged kubeconfig settings or a specified kubeconfig file
```

Certificates in KubeConfig

```yaml
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
```

Certificates in KubeConfig

```yaml
apiVersion: v1
kind: Config

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

    certificate-authority-data:
```


```
-----BEGIN CERTIFICATE -----
MIICWDCCAUACAQAwEzERMA8GA1UEAwwIbmV3LXVzZXIwggEiMA0G
AQUAA4IBDwAwggEKAoIBAQDO0WJW+DXsAJSIrjpNo5vRIBplnzg+
LfC27t+1eEnON5Muq99NevmMEOnrDUO/thyVqP2w2XNIDRXjYyF4
y3BihhB93MJ7Oql3UTvZ8TELqyaDknRl/jv/SxgXkok0ABUTpWMx
IF5nxAttMVkDPQ7NbeZRG43b+QWlVGR/z6DWOfJnbfezOtaAydGL
EcCXAwqChjBLkz2BHPR4J89D6Xb8k39pu6jpyngV6uP0tIbOzpqN
j2qEL+hZEWkkFz80lNNtyT5LxMqENDCnIgwC4GZiRGbrAgMBAAGg
9w0BAQsFAAOCAQEAS9iS6C1uxTuf5BBYSU7QFQHUzalNxAdYsaORF
hOK4a2zyNyi44OOijyaD6tUW8DSxkr8BLK8Kg3srREtJql5rLZy9
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF53lesNSNMLQ2++RMnjDQJ
Wr2EUM6UawzykrdHImwTv2mlMY0R+DNtV1Yie+0H9/YElt+FSGjh
4l3E/y3qL71WfAcuH3OsVpUUnQISMdQs0qWCsbE56CC5DhPGZIpU
vwQ07jG+hpknxmuFAeXxgUwodALaJ7ju/TDIcw==
-----END CERTIFICATE -----
```


```
cat ca.crt | base64
```


```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVN
tLS0KTUlJQ1dEQ0NBVUFDQVFBd0V6RVJNQThHQT
F3d0libVYzTFhWelpYSXdnZ0VpTUEwR0NTcUdTS
FFFQgpBUVVBQTRJQkR3QXdnZ0VLQW9JQkFRRE8w
K0RYc0FKU0lyanBObzV2UklCcGxuemcrNnhjOSt
rS2kwCkxmQzI3dCsxZUVuT041TXVxOTlOZXZtTU
J
```

Practice- cube config

Saturday, June 29, 2024 9:54 AM

Task Hint 58:59

1 2 3 4 5 6 7 8 9 10 11 12 13 14

Where is the default kubeconfig file located in the current environment?
Find the current home directory by looking at the HOME environment variable.

/root/kubeconfig

/root/.kube/config

/root/.kube/kubeconfig

/home/packer/.kube/config

Terminal 1

```
root@controlplane ~ ➜

root@controlplane ~ ➜  echo $HOME
/root

root@controlplane ~ ➜  pwd
/root

root@controlplane ~ ➜

root@controlplane ~ ➜  ls .kube/
cache  config

root@controlplane ~ ➜  ls .kube/config
.kube/config

root@controlplane ~ ➜  cat .kube//config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMvakNDQWVhZ0F3SUJBZ0lCQURBTkJn
a3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJeU1EUXhOekl5TWpVeU5Wb1hEVE15TURR
eE5ESXlNalV5TlZvd0ZURVRNQkVHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBREND
QVFvQ2dnRUJBT2hDCkUyb3hBdmFlRDNVU3hHOUNGa1pFdmZKM3Y1Q0NKbXkzOWVKZWFoOTNXVTRER1VaZmg3amMwWjh3UUhYUmIrYjIK
MSt5cWEvNldMeHQ5cXRSc2JFc0RKZ0ZzdFdsV3R1aExkCkZSSXFONzI4ZlVZRXlWcjJTQnhRUG9tRS9YK21qRwpwMnJERVpPV0Nsd1Rr
WHpKOUFpZXMrQnhVSlNHRVRzWTdhYy95Mm9ocEVqU0UyaWdPQUMxU3pKaWZCRy90aWtMCjlWLzNUUjNSa05VMXJ6NHBwMjhieFlIQ0cv
d0lCMlExTUhpVCtkaUhhTHRTDJubnZQL2hwL1B6U0k2cHZ3bk8KT0hqa2xzQUtMWUgrV2Ywak1VTjB3cFVwdnQvY0JOT3hTU0ZRUDdm
YTZkdXp5U3MwaXkrMnJDZHl6akFIeXdUcAptbWwxMUl5Q0NoVjJOQ1lYVkE4Q0F3RUFBYU5aTUZjd0RnWURWUjBQQVFIL0JBUURBZ0tr
TUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0hRWURWUjBPQkJZRUZBYy9jeHQyVWhLdk05aHA2eDU0TTJQbEc4RVdNQlVHQTFVZEVRUU8K
TUF5Q0NtdDFZbVZ5Ym1WMFpYTXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnRUJBQStQQlREQS9lREpzYXBXRHFaWApyQnd4UnpUcjh4UHcy
bHhzNWFTUGcxNEhDc0tTVkRsdld1M1E3dnVJaXFldWxnOVg4dE0rcXJSMkN4OUU2TDQ3CkpzRE1BVG9mUnJFL2dEQmY3ZWNSbTREWmYx
VUxjZzRYVUVXSE52UEFsOGRWOWhjVXZ0V0hzNFB4aUhWZjZnencKZENUd3NCNGZ0blExUE94TUlTTnk4c1MrWHY5eHN4SFFnRTBLaTZS
MXVyNTlHaE8vbmxNaHB4dUdreEJBQzJEUQpnZWdBaFkyYm90L0dYM1hvN2VkRVBtZjI5MWVZRnBXaTZVdkttcENqMnNhRGdFNWZDU3oy
ajRGMGxveDhUcmxVCk5HK1JvanoyZERnSWsrQTlDSnFKOE1xVGo1aDBYem04STR0YXhBbDN2YVIxRktzYnd4YzllSkZUQmZKTFk3ZEIK
SlJNPQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://controlplane:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

How many contexts are defined in the default kubeconfig file?

```
YTZkdXp5U3MwaXkrMnJDZHl6akFIeXdUcAptbWwxMUl5Q0NoVjJOQ1lYVkE4Q0F3RUFBYU5aTUZjd0RnWURWUjBQQVFIL0JBUURBZ0tr
TUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0hRWURWUjBPQkJZRUZBYy9jeHQyVWhLdk05aHA2eDU0TTJQbEc4RVdNQlVHQTFVZEVRUU8K
TUF5Q0NtdDFZbVZ5Ym1WMFpYTXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnRUJBQStQQlREQS9lREpzYXBXRHFaWApyQnd4UnpUcjh4UHcy
bHhzNWFTUGcxNEhDc0tTVkRsdld1M1E3dnVJaXFldWxnOVg4dE0rcXJSMkN4OUU2TDQ3CkpzRE1BVG9mUnJFL2dEQmY3ZWNSbTREWmYx
VUxjZzRYVUVXSE52UEFsOGRWOWhjVXZ0V0hzNFB4aUhWZjZnencKZENUd3NCNGZ0blExUE94TUlTTnk4c1MrWHY5eHN4SFFnRTBLaTZS
MXVyNTlHaE8vbmxNaHB4dUdreEJBQzJEUQpnZWdBaFkyYm90L0dYM1hvN2VkRVBtZjI5MWVZRnBXaTZVdkttcENqMnNhRGdFNWZDU3oy
ajRGMGxveDhUcmxVCk5HK1JvanoyZERnSWsrQTlDSnFKOE1xVGo1aDBYem04STR0YXhBbDN2YVIxRktzYnd4YzllSkZUQmZKTFk3ZEIK
SlJNPQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://controlplane:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURJVENDQWdtZ0F3SUJBZ0lJV0ZYUGJGZVN
lc1l3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp1WlhSbGN6QWVGdzB5TWpBME1UY3lNakkxTWpWYUZ
3MHlNekEwTVRjeU1qSTFNamhhTURReApGekFWQmdOVkJBb1REbk41YzNSbGJUcHRZWE4wWlhKek1Sa3dGd1lELVlFRREV4QnJkV0psY20
1bGRHVnpMV0ZrCmJXbHVNSUlCSWpBTkJna3Foa2lHOXcwQkFRRUZBQU9DQVE4QU1JSUJDZ0tDQVFFQXdFN3Rmck9yZ3RGZ20yRmsKUlJ
nMk83TEw0dk5QMWRwcnZFdldKUEw2Yy9BUXVPTXRVOC920HRha2syWWVkdUZDRG9EcFJoU3dyQmVsU0daUwppQWdyMExhUk45L293TGx
FZ3ZiZzlmb3pZZnZQQzBuREdUdlcxek96ZHFNN0JLU3RBOTZaUWR4UytJSHFwWlpuCmhUa0hzNHJocFJBbGROVTdyaHkydml4L0VtSEJ
0alYwN0hScUdVc2QyNG5LK29YVWdCeHpRelEvNUM4UHhKWW0KUWhuaUNiRDkyRDZaeXFaWlFJV3F0YlZVSkp1endVUWIrcUJ3WnVjTlF
KT3ZvNTJ2b1d2aTFNK2dQTFlhVkVvRQo5ZW51eUlncUJSMVlPOHVvZGlyM1RHVWJtUFhHVUU4c2twNjBMaUE2WU9xQ3lwdlV1WHJVblV
YRmo2V3ZmRnV0Ci8rT01JUUlEQVFBQm8xWXdWREFPQmdOVkhROEJBZjhFQkFNQ0JhQXdFd1lEVlIwbEJBd3dDZ1lJS3dZQkJRVUgKQXd
Jd0RBWURWUjBUQVFIL0JBSXdBREFmQmdOVkhTTUVHREFXZ0JRSFAzTWJkbElTcnpQWWFlc2VlRE5qNVJ2QgpGakFOQmdrcWhraUc5dzB
CQVFzRkFBT0NBUUVBdlF1bkh4VVpSNkxmeVNkd0tHcTJ2SVdKejAwVXdRNmZzRjA5CkVDR2xrT3pCNURZVlozcVovYzcrQVhNM1Bqc3M
1a2ljOFlicGFuU3JjZDMwZFFxZENabWM4b0h0QW1JcEc3dW4KSDA4U1orVndCejR0NWNwQjRJMlNZenRIWGY5amN6Q3V6dnFqRVRuWS9
rUnQxY0tjZm9xNkpZejk0Ulc4MEpnNgpvOUtVRjlCQzFGT1lFMENpVHNlT013cXgxSThLYjI5L1Z6QWE4LzRSaDBYK2Y2aUw5S0Jtem1
Lay9iWUVPSkdQCitWTzFWRzhZbWg5MUYvSElXdkNZUjFKUUMrVUNrOFNWK2JnT2dQTW8vbFkrZlNqYXBhSUN0K2hZbjlFSWY1cEYKbUN
NZkh0Q29GcmVLY1lucVFGcjZMSzh4NlhxNjc5b0orc1RBdElTeUFaSkdUM2pTMGc9PQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg=
=
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

What is the user configured in the current context?

- root
- admin
- sysadmin
- kube-admin
- kubernetes-admin

```
YTZkdXp5U3MwaXkrMnJDZHl6akFIeXdUcAptbWwxMUl5Q0NoVjJOQ1lYVkE4Q0F3RUFBYU5aTUZjd0RnWURWUjBQQVFIL0JBUURBZ0tr
TUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0hRWURWUjBPQkJZRUZBYy9jeHQyVWhLdk05aHA2eDU0TTJQbEc4RVdNQlVHQTFVZEVRUU8K
TUF5Q0NtdDFZbVZ5Ym1WMFpYTXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnRUJBQStQQlREQS9lREpzYXBXRHFaWApyQnd4UnpUcjh4UHcy
bHhzNWFTUGcxNEhDc0tTVkRsdld1M1E3dnVJaXFldWxnOVg4dE0rcXJSMkN4OUU2TDQ3CkpzRE1BVG9mUnJFL2dEQmY3ZWNSbTREWmYx
VUxjZzRYVUVXSE52UEFsOGRWOWhjVXZ0V0hzNFB4aUhWZjZnencKZENUd3NCNGZ0blExUE94TUlTTnk4c1MrWHY5eHN4SFFnRTBLaTZS
MXVyNTlHaE8vbmxNaHB4dUdreEJBQzJEUQpnZWdBaFkyYm90L0dYM1hvN2VkRVBtZjI5MWVZRnBXaTZVdkttcENqMnNhRGdFNWZDU3oy
ajRGMGxveDhUcmxVCk5HK1JvanoyZERnSWsrQTlDSnFKOE1xVGo1aDBYem04STR0YXhBbDN2YVIxRktzYnd4YzllSkZUQmZKTFk3ZEIK
SlJNPQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://controlplane:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURJVENDQWdtZ0F3SUJBZ0lJV0ZYUGJGZVN
lc1l3RFFZSktvWkldmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp1WlhSbGN6QWVGdzB5TWpBME1UY3lNakkxTWpWYUZ
3MHlNekEwTVRjeU1qSTFNamhhTURReApGekFWQmdOVkJBb1REbk41YzNSbGJUcHRZWE4wWlhKek1Sa3dGd1lEVlFRREV4QnJkV0psY20
1bGRHVnpMV0ZrCmJXbHVNSUlCSWpBTkJna3Foa2lHOXcwQkFRRUZBQU9DQVE4QU1JSUJDZ0tDQVFFQXdFN3Rmck9yZ3RGZ20yRmsKUlJ
nMk83TEw0dk5QMWRwcnZFdldKUEw2Yy9BUXVPTXRVOC92OHRha2syWWVkdUZDRG9EcFJoU3dyQmVsU0daUwppQWdyMExhUk45L293TGx
FZ3ZiZzlmb3pZZnZQQzBuREdUdlcxek96ZHFNN0JLU3RBOTZaUWR4UytJSHFwWlpuCmhUa0hzNHJocFJBbGROVTdyaHkydml4L0VtSEJ
0alYwN0hScUdVc2QyNG5LK29YVWdCeHpRelEvNUM4UHhKWW0KUWhuaUNiRDkyRDZaeXFaWlFJV3F0YlZVSkp1endVUWIrcUJ3WnVjTlF
KT3ZvNTJ2b1d2aTFNK2dQTFlhVkVvRQo5ZW51eUlncUJSMVlPOHVvZGlyM1RHVWJtUFhHVUU4c2twNjBMaUE2WU9xQ3lwdlV1WHJVblV
YRmo2V3ZmRnV0Ci8rT01JUUlEQVFBQm8xWXdWREFPQmdOVkhROEJBZjhFQkFNQ0JhQXdFd1lEVlIwbEJBd3dDZ1lJS3dZQkJRVUgKQXd
Jd0RBWURWUjBUQVFIL0JBSXdBREFmQmdOVkhTTUVHREFXZ0JRSFAzTWJkbElTcnpQWWFlc2VlRE5qNVJ2QgpGakFOQmdrcWhraUc5dzB
CQVFzRkFBT0NBUUVBdlF1bkh4VVpSNkxmeVNkd0tHcTJ2SVdKejAwVXdRNmZzRjA5CkVDR2xrT3pCNURZVlozcVovYzcrQVhNM1Bqc3M
1a2ljOFlicGFuU3JjZDMwZFFxZENabWM4b0h0QW1JcEc3dW4KSDA4U1orVndCejR0NWNwQjRJMlNZenRIWGY5amN6Q3V6dnFqRVRuWS9
rUnQxY0tjZm9xNkpZejk0Ulc4MEpnNgpvOUtVRjlCQzFGT1lFMENpVHNlT013cXgxSThLYjI5L1Z6QWE4LzRSaDBYK2Y2aUw5S0Jtem1
Lay9iWUVPSkdQCitWTzFWRzhZbWg5MUYvSElXdkNZUjFKUUMrVUNrOFNWK2JnT2dQTW8vbFkrZlNqYXBhSUN0K2hZbjlFSWY1cEYKbUN
NZkh0Q29GcmVLY1lucVFGcjZMSzh4NlhxNjc5b0orc1RBdElTeUFaSkdUM2pTMGc9PQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg=
=
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

What is the name of the cluster configured in the default kubeconfig file?

- kube
- kube-cluster
- kubernetes
- default-cluster

```
root@controlplane ~ ➜  ls .kube/config
.kube/config

root@controlplane ~ ➜  cat .kube//config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMvakNDQWVhZ0F3SUJBZ0lCQURBTkJn
a3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJeU1EUXhOekl5TWpVeU5Wb1hEVE15TURR
eE5ESXlNalV5TlZvd0ZURVRNQkVHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBREND
QVFvQ2dnRUJBT2hDCkUyb3hBdmFlRDNVU3hHOUNGa1pFdmZKM3Y1Q0NKbXkzOWVKZWFoOTNXVTRER1VaZmg3amMwWjh3UUhYUmIrYjIK
MSt5cWEvNldMeHQ5cXRSc2JFc0RKZ0ZzdFdsV3R1aExkckZSSXFONzI4ZlVZRXlWcjJTQnhRUG9tRS9YK21qRwpwMnJERVpPV0Nsd1Rr
WHpKOUFpZXMrQnhVSlNHRVRzWTdhYy95Mm9ocEVqU0UyaWdPQUMxU3pKaWZCRy90aWtMCjlWLzNUUjNSa05VMXJ6NHBwMjhieFlIQ0cv
d0lCMlExTUhpVCtkaUhhTHRRTDJubnZQL2hwL1B6U0k2cHZ3bk8KT0hqa2xzQUtMWUgrV2Ywak1VTjB3cFVwdnQvY0JOT3hTU0ZRUDdm
YTZkdXp5U3MwaXkrMnJDZHl6akFIeXdUcAptbWwxMUl5Q0NoVjJOQ1lYVkE4Q0F3RUFBYU5aTUZjd0RnWURWUjBQQVFIL0JBUURBZ0tr
TUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0hRWURWUjBPQkJZRUZBYy9jeHQyVWhLdk05aHA2eDU0TTJQbEc4RVdNQlVHQTFVZEVRUU8K
TUF5Q0NtdDFZbVZ5Ym1WMFpYTXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnRUJBQStQQlREQS9lREpzYXBXRHFaWApyQnd4UnpUcjh4UHcy
bHhzNWFTUGcxNEhDc0tTVkRsdld1M1E3dnVJaXFldWxnOVg4dE0rcXJSMkN4OUU2TDQ3CkpzRE1BVG9mUnJFL2dEQmY3ZWNSbTREWmYx
VUxjZzRYVUVXSE52UEFsOGRWOWhjVXZ0V0hzNFB4aUhWZjZnencKZENUd3NCNGZ0blExUE94TUlTTnk4c1MrWHY5eHN4SFFnRTBLaTZS
MXVyNTlHaE8vbmxNaHB4dUdreEJBQzJEUQpnZWdBaFkyYm90L0dYM1hvN2VkRVBtZjI5MWVZRnBXaTZVdkttcENqMnNhRGdFNWZDU3oy
ajRGMGxveDhUcmxVCk5HK1JvanoyZERnSWsrQTlDSnFKOE1xVGo1aDBYem04STR0YXhBbDN2YVIxRktzYnd4YzllSkZUQmZKTFk3ZEIK
SlJNPQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://controlplane:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
```

Task Hint 56:31

1 2 3 4 5 6 7 8 9 10 11 12 13 14

A new kubeconfig file named `my-kube-config` is created. It is placed in the `/root` directory. How many clusters are defined in that kubeconfig file?

1

2

3

4

5

Terminal 1 +

```
root@controlplane ~ ➜  cat /root/my-kube-config
apiVersion: v1
kind: Config

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

- name: development
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://controlplane:6443

- name: kubernetes-on-aws
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://controlplane:6443

- name: test-cluster-1
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://controlplane:6443

contexts:
- name: test-user@development
  context:
    cluster: development
    user: test-user
```

Task 56:19

1 2 3 4 5 6 7 8 9 10 11 12 13 14

How many contexts are configured in the `my-kube-config` file?

1

2

3

4

5

Terminal 1 +

```
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://controlplane:6443

- name: kubernetes-on-aws
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://controlplane:6443

- name: test-cluster-1
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://controlplane:6443

contexts:
- name: test-user@development
  context:
    cluster: development
    user: test-user

- name: aws-user@kubernetes-on-aws
  context:
    cluster: kubernetes-on-aws
    user: aws-user

- name: test-user@production
  context:
    cluster: production
    user: test-user

- name: research
  context:
    cluster: test-cluster-1
    user: dev-user
```

Task 56:05

1 2 3 4 5 6 7 8 9 10 11 12 13 14

What user is configured in the `research` context?

aws-user

test-user

admin-user

dev-user

Terminal 1

```
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://controlplane:6443

- name: kubernetes-on-aws
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://controlplane:6443

- name: test-cluster-1
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://controlplane:6443

contexts:
- name: test-user@development
  context:
    cluster: development
    user: test-user

- name: aws-user@kubernetes-on-aws
  context:
    cluster: kubernetes-on-aws
    user: aws-user

- name: test-user@production
  context:
    cluster: production
    user: test-user

- name: research
  context:
    cluster: test-cluster-1
    user: dev-user

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

1 2 3 4 5 6 7 8 9 10 11 12 13 14

What is the name of the client-certificate file configured for the `aws-user` ?

aws-user.crt

aws-user.key

aws-certificate.crt

ca.crt

```
contexts:
- name: test-user@development
  context:
    cluster: development
    user: test-user

- name: aws-user@kubernetes-on-aws
  context:
    cluster: kubernetes-on-aws
    user: aws-user

- name: test-user@production
  context:
    cluster: production
    user: test-user

- name: research
  context:
    cluster: test-cluster-1
    user: dev-user

users:
- name: test-user
  user:
    client-certificate: /etc/kubernetes/pki/users/test-user/test-user.crt
    client-key: /etc/kubernetes/pki/users/test-user/test-user.key
- name: dev-user
  user:
    client-certificate: /etc/kubernetes/pki/users/dev-user/developer-user.crt
    client-key: /etc/kubernetes/pki/users/dev-user/dev-user.key
- name: aws-user
  user:
    client-certificate: /etc/kubernetes/pki/users/aws-user/aws-user.crt
    client-key: /etc/kubernetes/pki/users/aws-user/aws-user.key

current-context: test-user@development
preferences: {}


root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

What is the current context set to in the `my-kube-config` file?

- aws-user@kubernetes
- research
- test-user@development
- test-user@production

```
contexts:
- name: test-user@development
  context:
    cluster: development
    user: test-user

- name: aws-user@kubernetes-on-aws
  context:
    cluster: kubernetes-on-aws
    user: aws-user

- name: test-user@production
  context:
    cluster: production
    user: test-user

- name: research
  context:
    cluster: test-cluster-1
    user: dev-user

users:
- name: test-user
  user:
    client-certificate: /etc/kubernetes/pki/users/test-user/test-user.crt
    client-key: /etc/kubernetes/pki/users/test-user/test-user.key
- name: dev-user
  user:
    client-certificate: /etc/kubernetes/pki/users/dev-user/developer-user.crt
    client-key: /etc/kubernetes/pki/users/dev-user/dev-user.key
- name: aws-user
  user:
    client-certificate: /etc/kubernetes/pki/users/aws-user/aws-user.crt
    client-key: /etc/kubernetes/pki/users/aws-user/aws-user.key

current-context: test-user@development
preferences: {}


root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

I would like to use the `dev-user` to access `test-cluster-1`. Set the current context to the right one so I can do that.
Once the right context is identified, use the `kubectl config use-context` command.

Check

● Current context set

```
- name: aws-user@kubernetes-on-aws
  context:
    cluster: kubernetes-on-aws
    user: aws-user

- name: test-user@production
  context:
    cluster: production
    user: test-user

- name: research
  context:
    cluster: test-cluster-1
    user: dev-user

users:
- name: test-user
  user:
    client-certificate: /etc/kubernetes/pki/users/test-user/test-user.crt
    client-key: /etc/kubernetes/pki/users/test-user/test-user.key
- name: dev-user
  user:
    client-certificate: /etc/kubernetes/pki/users/dev-user/developer-user.crt
    client-key: /etc/kubernetes/pki/users/dev-user/dev-user.key
- name: aws-user
  user:
    client-certificate: /etc/kubernetes/pki/users/aws-user/aws-user.crt
    client-key: /etc/kubernetes/pki/users/aws-user/aws-user.key

current-context: test-user@development
preferences: {}


root@controlplane ~ ➜  kubectl config use-context research --kubeconfig /root/my-kube-config
Switched to context "research".

root@controlplane ~ ➜

root@controlplane ~ ➜  cat /root/my-kube-config
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

We don't want to have to specify the kubeconfig file option on each command. Make the `my-kube-config` file the default kubeconfig.

Check

- Default kubeconfig file configured

```
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://controlplane:6443
  name: test-cluster-1
contexts:
- context:
    cluster: kubernetes-on-aws
    user: aws-user
  name: aws-user@kubernetes-on-aws
- context:
    cluster: test-cluster-1
    user: dev-user
  name: research
- context:
    cluster: development
    user: test-user
  name: test-user@development
- context:
    cluster: production
    user: test-user
  name: test-user@production
current-context: research
kind: Config
preferences: {}
users:
- name: aws-user
  user:
    client-certificate: /etc/kubernetes/pki/users/aws-user/aws-user.crt
    client-key: /etc/kubernetes/pki/users/aws-user/aws-user.key
- name: dev-user
  user:
    client-certificate: /etc/kubernetes/pki/users/dev-user/developer-user.crt
    client-key: /etc/kubernetes/pki/users/dev-user/dev-user.key
- name: test-user
  user:
    client-certificate: /etc/kubernetes/pki/users/test-user/test-user.crt
    client-key: /etc/kubernetes/pki/users/test-user/test-user.key

root@controlplane ~ ➜  mv /root/my-kube-config /root/.kube/config

root@controlplane ~ ➜  cat /root/.kube/
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

With the current-context set to `research`, we are trying to access the cluster. However something seems to be wrong. Identify and fix the issue.
Try running the `kubectl get pods` command and look for the error. All users certificates are stored at `/etc/kubernetes/pki/users`.

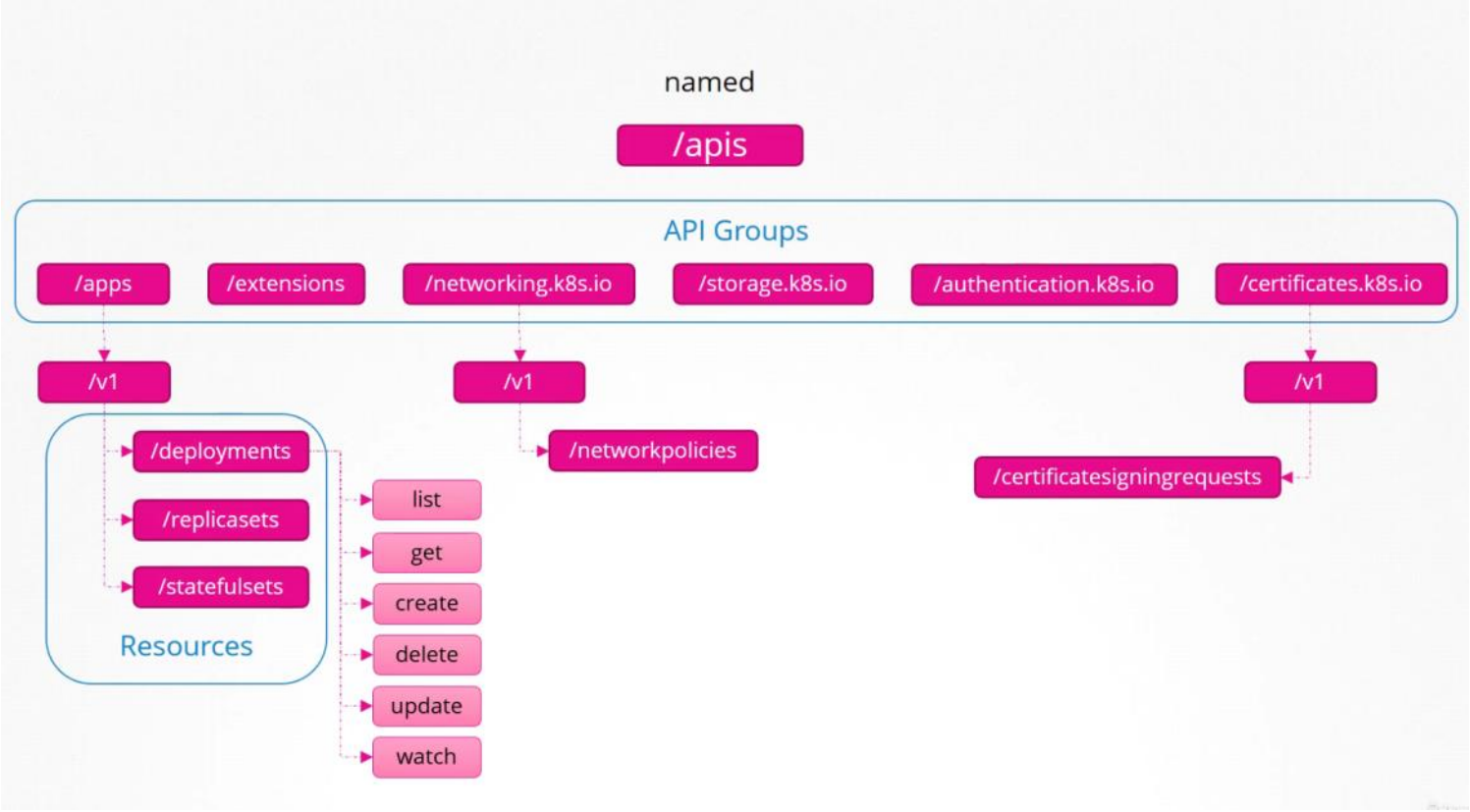
Check

- Issue fixed

```
- context:
    cluster: kubernetes-on-aws
    user: aws-user
  name: aws-user@kubernetes-on-aws
- context:
    cluster: test-cluster-1
    user: dev-user
  name: research
- context:
    cluster: development
    user: test-user
  name: test-user@development
- context:
    cluster: production
    user: test-user
  name: test-user@production
current-context: research
kind: Config
preferences: {}
users:
- name: aws-user
  user:
    client-certificate: /etc/kubernetes/pki/users/aws-user/aws-user.crt
    client-key: /etc/kubernetes/pki/users/aws-user/aws-user.key
- name: dev-user
  user:
    client-certificate: /etc/kubernetes/pki/users/dev-user/developer-user.crt
    client-key: /etc/kubernetes/pki/users/dev-user/dev-user.key
- name: test-user
  user:
    client-certificate: /etc/kubernetes/pki/users/test-user/test-user.crt
    client-key: /etc/kubernetes/pki/users/test-user/test-user.key

root@controlplane ~ ➜  ls /etc/kubernetes/pki/users
aws-user  dev-user  test-user

root@controlplane ~ ➜  ls /etc/kubernetes/pki/users/dev-user/
dev-user.crt  dev-user.csr  dev-user.key

root@controlplane ~ ➜
```

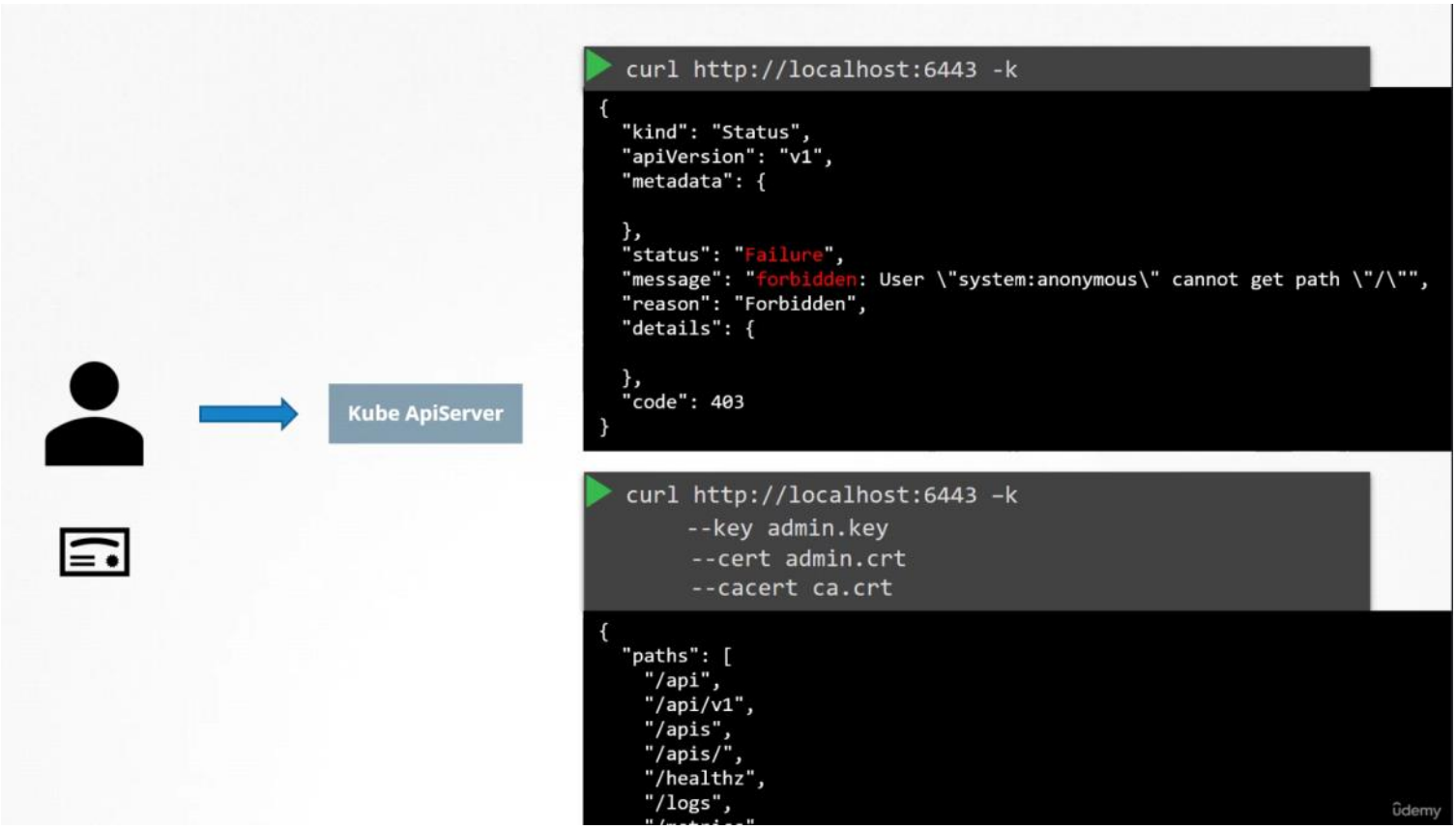
API groups

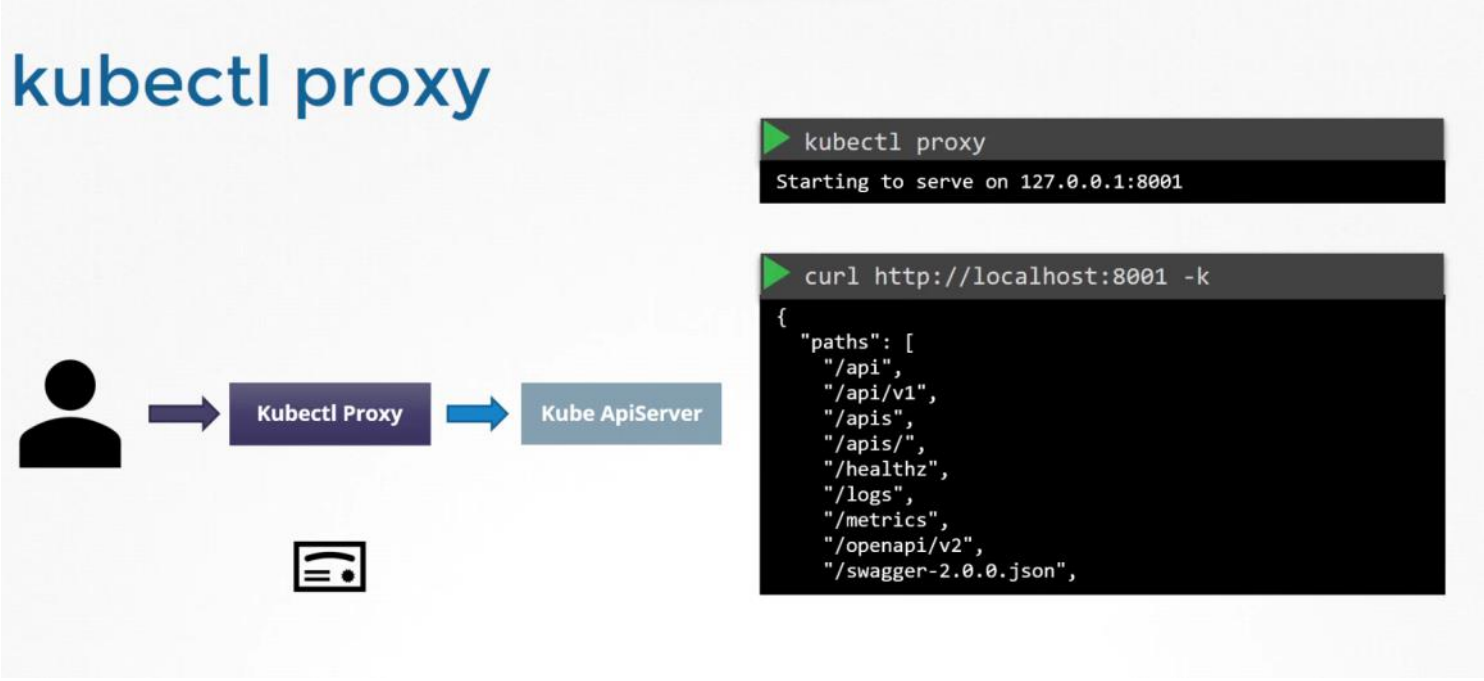
Saturday, June 29, 2024 10:13 AM



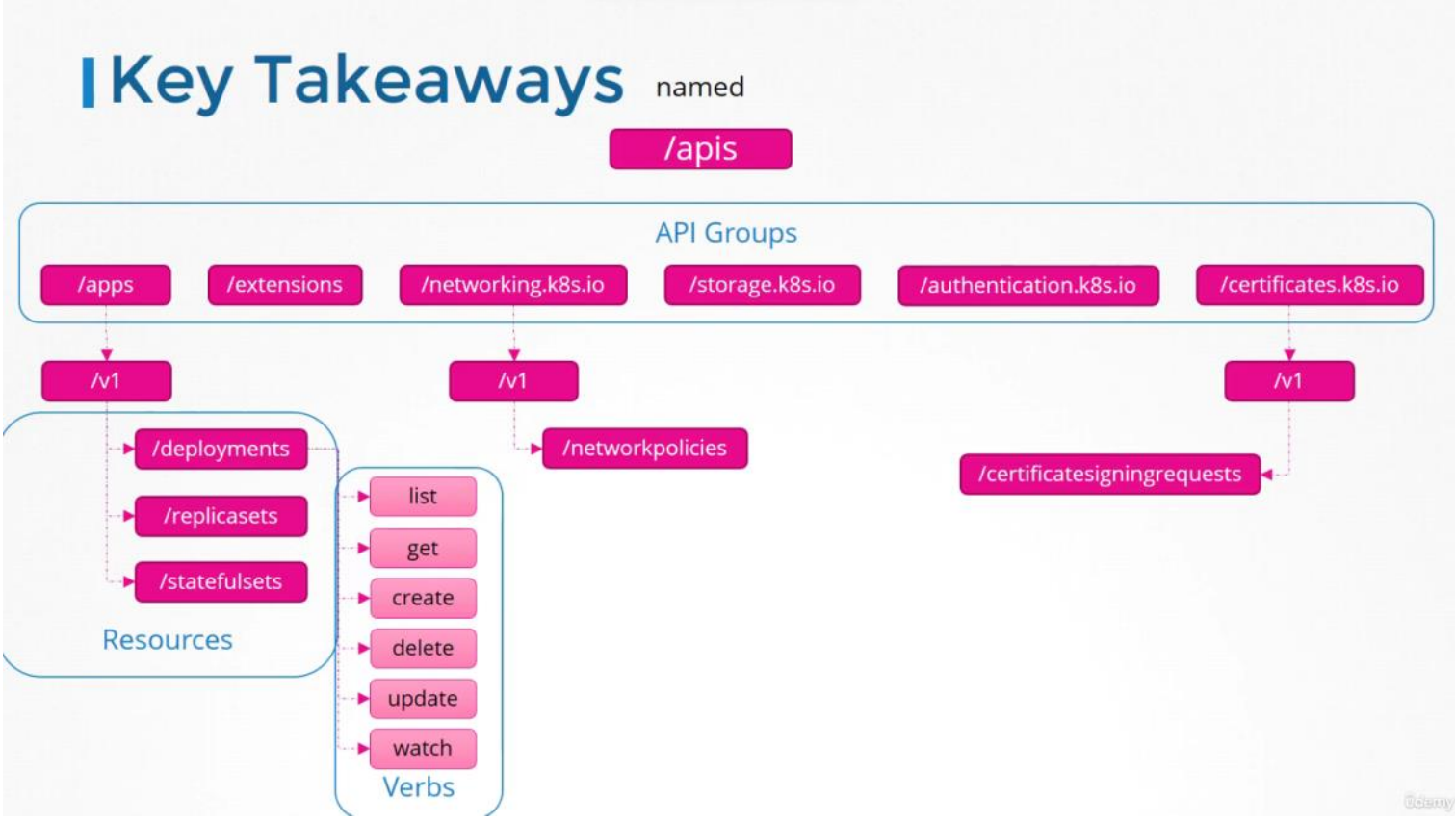

```
curl http://localhost:6443 -k
{
  "paths": [
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/healthz",
    "/logs",
    "/metrics",
    "/openapi/v2",
    "/swagger-2.0.0.json",
```

```
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",
```


Kube ApiServer
curl http://localhost:6443 -k
{
"kind": "Status",
"apiVersion": "v1",
"metadata": {
},
"status": "Failure",
"message": "forbidden: User \"system:anonymous\" cannot get path \"/\"",
"reason": "Forbidden",
"details": {
},
"code": 403
}
curl http://localhost:6443 -k
--key admin.key
--cert admin.crt
--cacert ca.crt
{
"paths": [
"/api",
"/api/v1",
"/apis",
"/apis/",
"/healthz",
"/logs",
Udemy

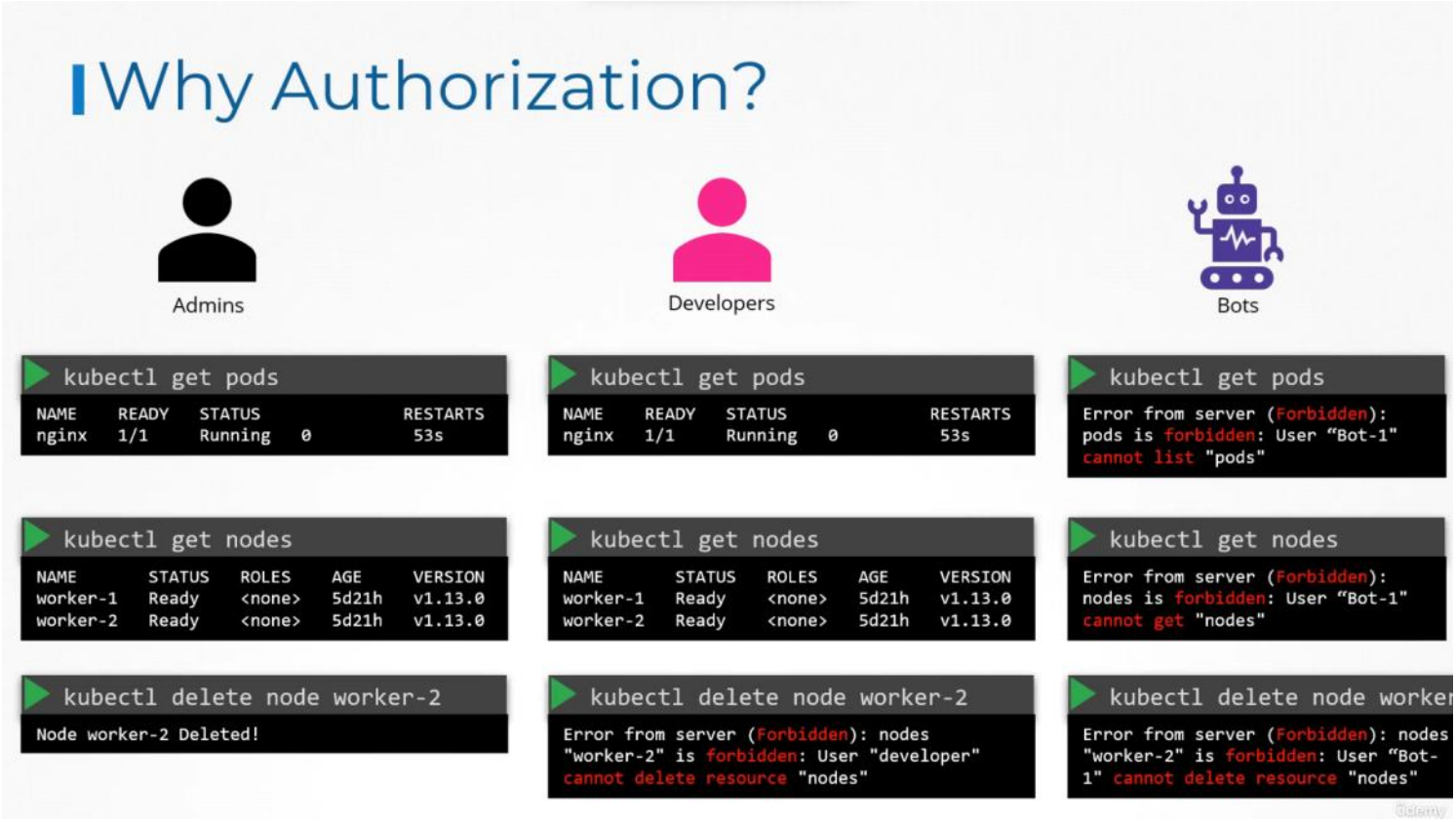

kubectl proxy
Kubectl Proxy
Kube ApiServer
kubectl proxy
Starting to serve on 127.0.0.1:8001
curl http://localhost:8001 -k
{
"paths": [
"/api",
"/api/v1",
"/apis",
"/apis/",
"/healthz",
"/logs",
"/metrics",
"/openapi/v2",
"/swagger-2.0.0.json",

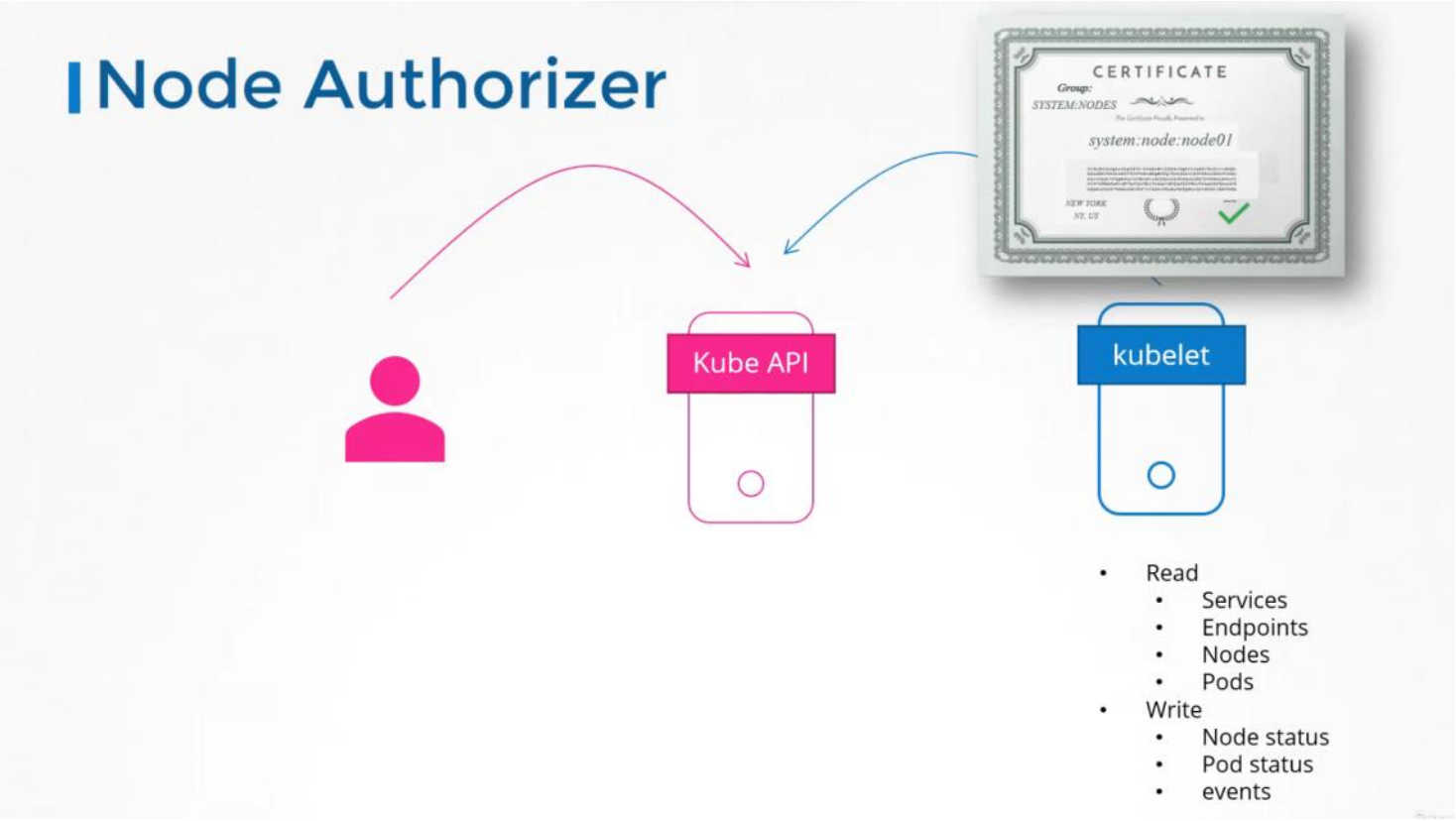
Kube proxy ≠ Kubectl proxy



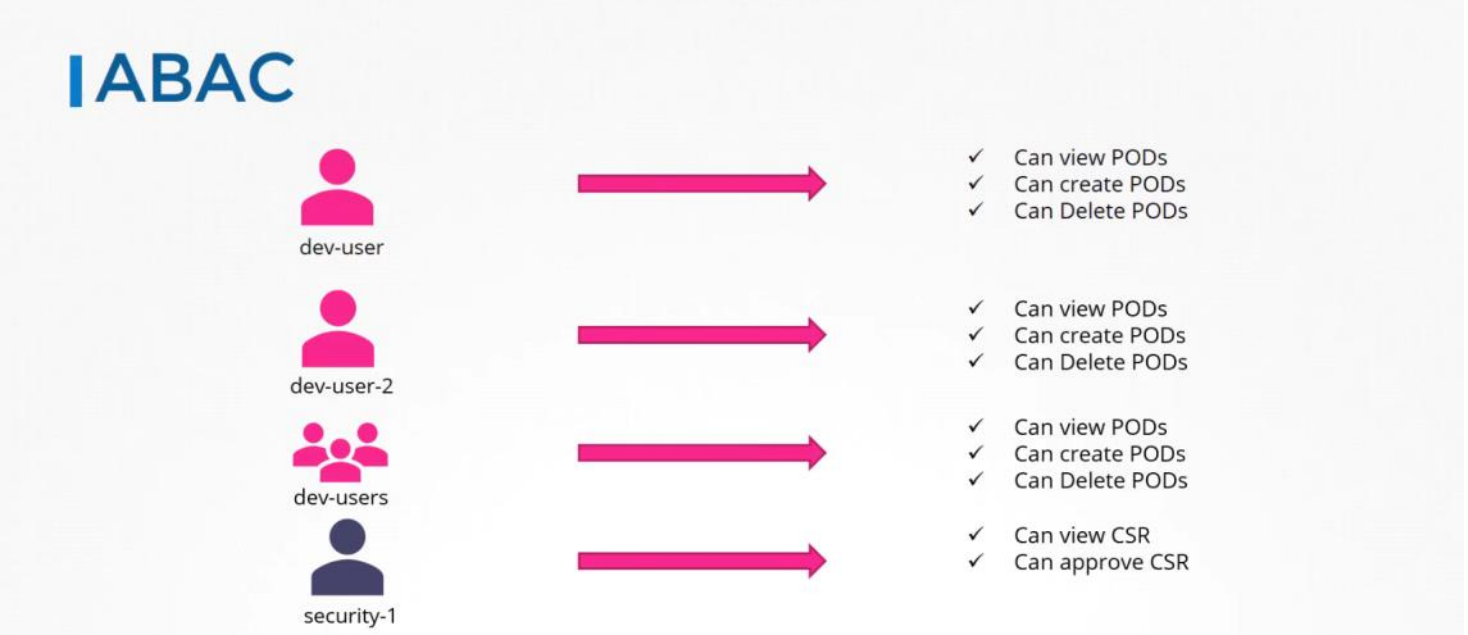
Authorization

Saturday, June 29, 2024 10:34 AM



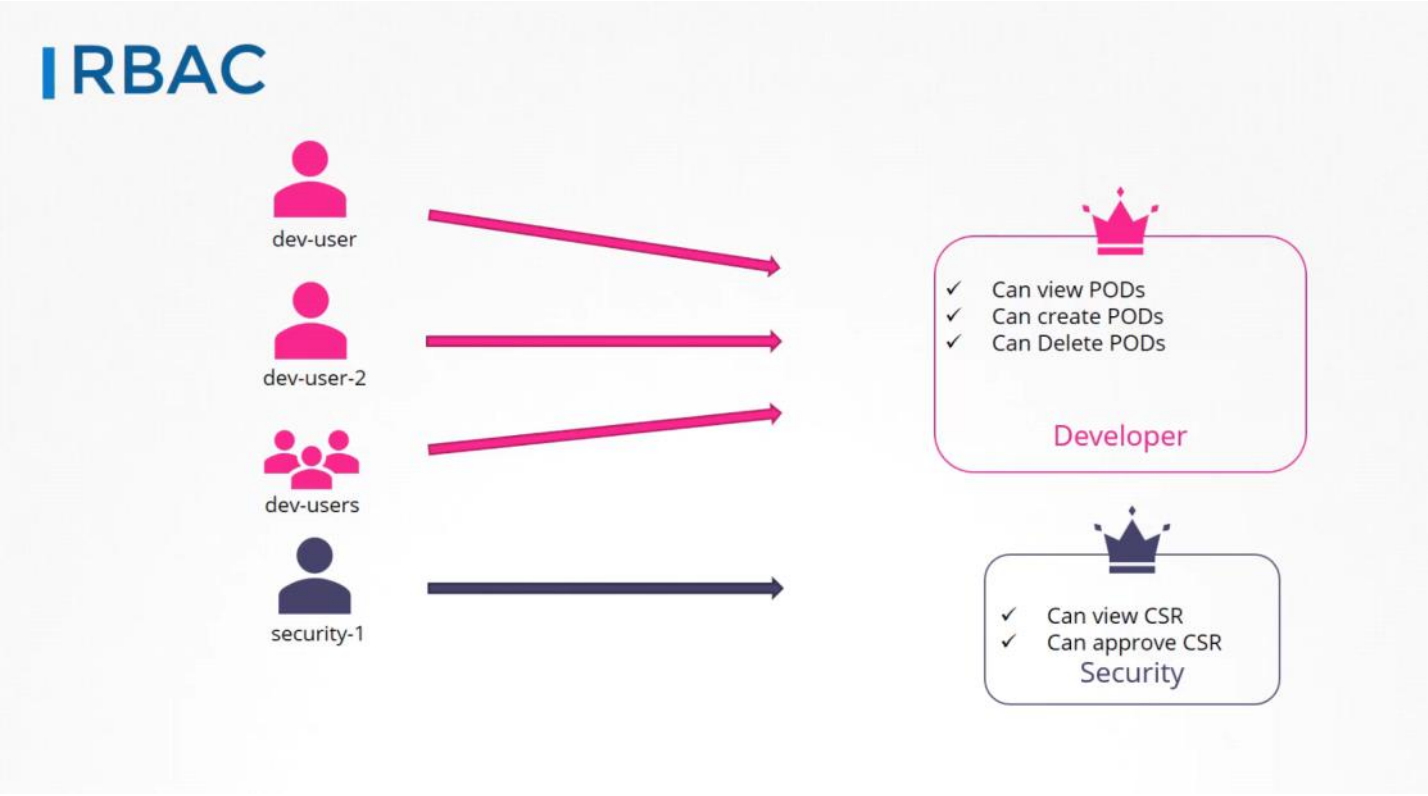


nde authirizer grants the permission here

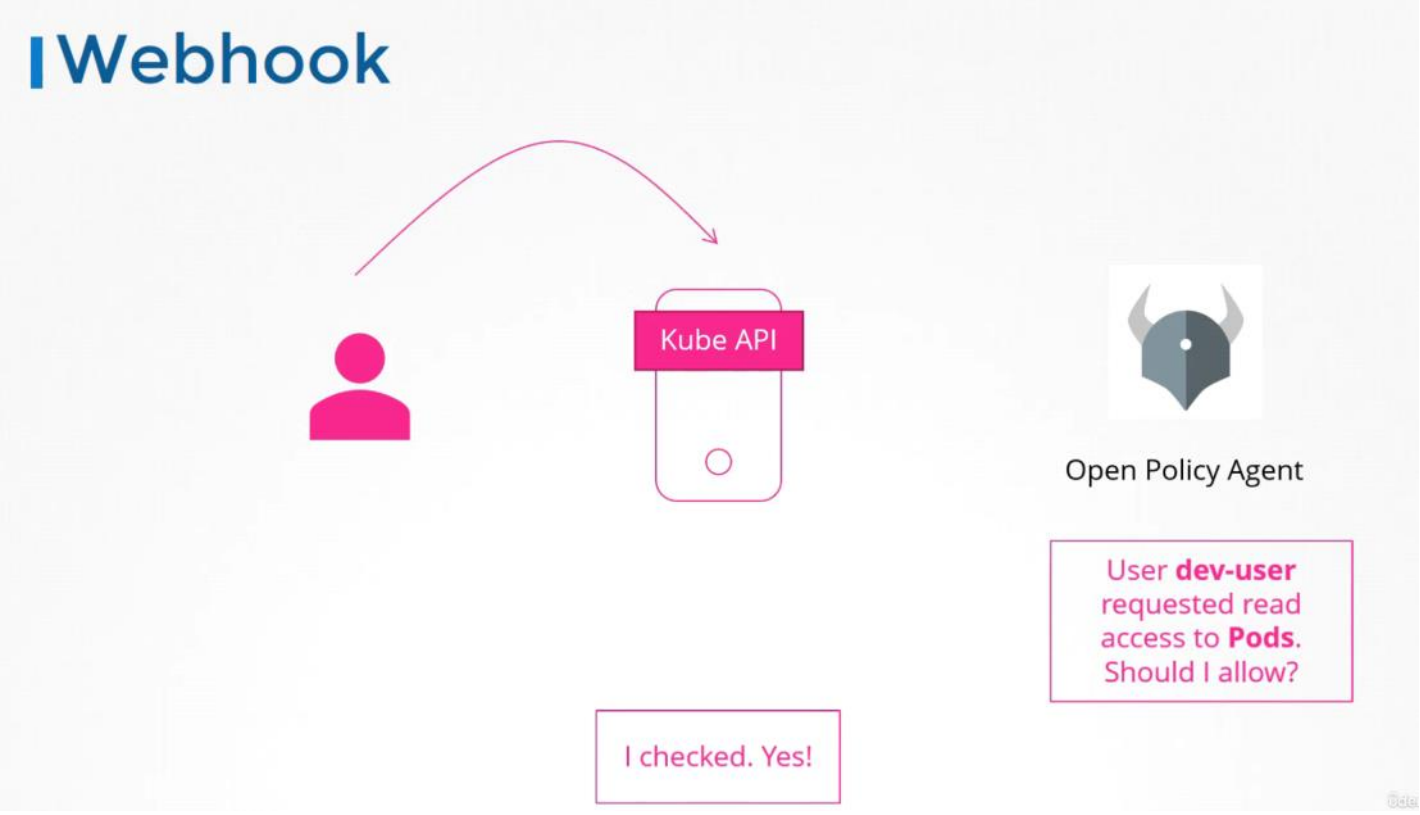



```
{"kind": "Policy", "spec": {"user": "dev-user", "namespace": "*", "resource": "pods", "apiGroup": "*"}}
{"kind": "Policy", "spec": {"user": "dev-user-2", "namespace": "*", "resource": "pods", "apiGroup": "*"}}
{"kind": "Policy", "spec": {"group": "dev-users", "namespace": "*", "resource": "pods",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

Attribute based access control defines a file in which you will define a policy file which contains all attributes of a user. each time you want to gramt permisison to a user you will have to edit this poliicy file and strat the api server.

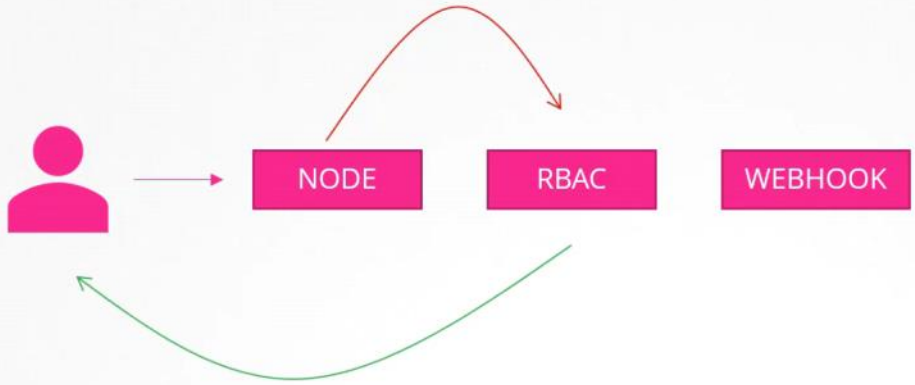


in role based access control we define a role with all permissions a developer needs an then associate all developers to this role. whenever changes are req we just modify the role and it reflects all the developers.



it uses a third party tool to grant access based on api callls it makes to the agent.

Authorization Mode




```
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 \\
```

by default this authorisation mode is set to always allow

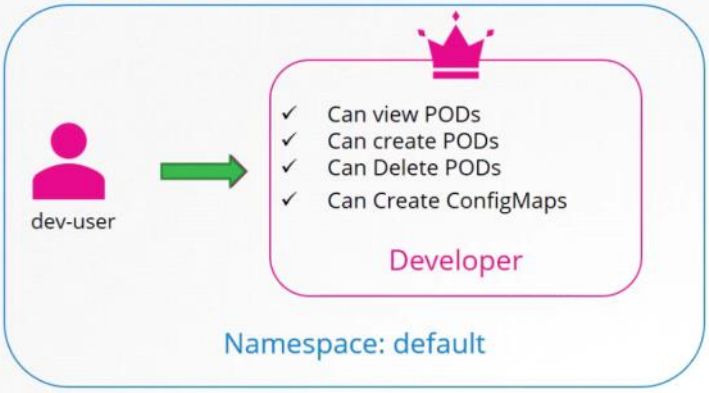
RBAC

developer-role.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["list", "get", "create", "update", "delete"]
- apiGroups: [""]
  resources: ["ConfigMap"]
  verbs: ["create"]
```

```
kubectl create -f developer-role.yaml
```

RBAC




```
kubectl create -f devuser-developer-binding.yaml
```

developer-role.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["list", "get", "create", "update", "de
- apiGroups: [""]
  resources: ["ConfigMap"]
  verbs: ["create"]
```

devuser-developer-binding.yaml

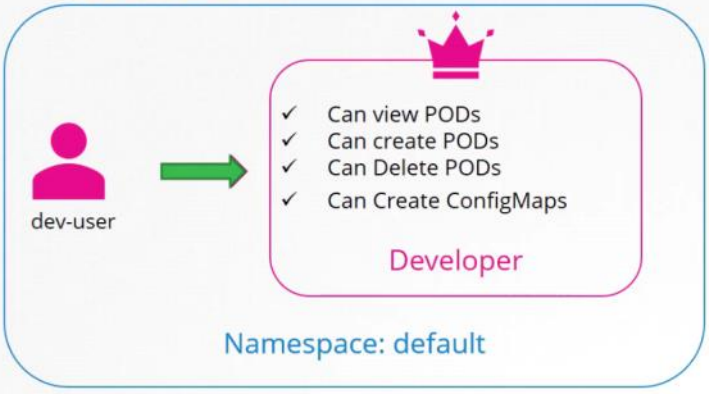
```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: devuser-developer-binding
subjects:
- kind: User
  name: dev-user
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: developer
  apiGroup: rbac.authorization.k8s.io
```

RBAC

Saturday, June 29, 2024 11:06 AM

Role based access control

RBAC




```
kubectl create -f devuser-developer-binding.yaml
```

developer-role.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["list", "get", "create", "update", "de
- apiGroups: [""]
  resources: ["ConfigMap"]
  verbs: ["create"]
```

devuser-developer-binding.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: devuser-developer-binding
subjects:
- kind: User
  name: dev-user
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: developer
  apiGroup: rbac.authorization.k8s.io
```

View 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]
```

View 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
```

Check Access

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

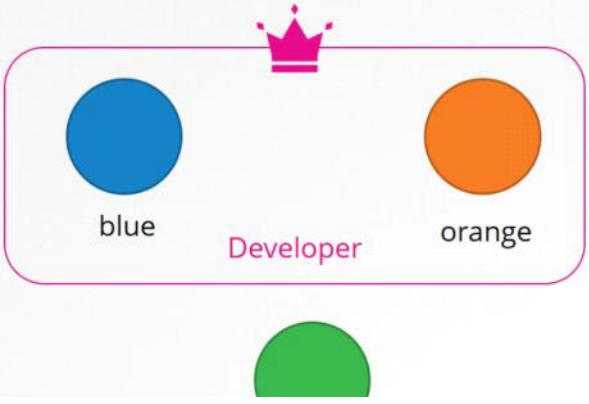
```
kubectl auth can-i delete nodes
no
```

```
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



developer-role.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "create", "update"]
```

you can redtrict the permissions by giving resource name

Practice- role based access control

Saturday, June 29, 2024 12:46 PM

1 2 3 4 5 6 7 8 9 10 11

Inspect the environment and identify the authorization modes configured on the cluster.

Check the `kube-apiserver` settings.

- Node
- RBAC
- ABAC
- Node,RBAC

```
root@controlplane ~ ➜

root@controlplane ~ ➜

root@controlplane ~ ➜  cat /etc/kubernetes/manifests/kube-apiserver.yaml
```

1 2 3 4 5 6 7 8 9 10 11

Inspect the environment and identify the authorization modes configured on the cluster.

Check the `kube-apiserver` settings.

- Node
- RBAC
- ABAC
- Node,RBAC

```
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=10.48.174.6
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
    - --requestheader-allowed-names=front-proxy-client
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
    - --requestheader-extra-headers-prefix=X-Remote-Extra-
    - --requestheader-group-headers=X-Remote-Group
    - --requestheader-username-headers=X-Remote-User
    - --secure-port=6443
    - --service-account-issuer=https://kubernetes.default.svc.cluster.local
    - --service-account-key-file=/etc/kubernetes/pki/sa.pub
    - --service-account-signing-key-file=/etc/kubernetes/pki/sa.key
    - --service-cluster-ip-range=10.96.0.0/12
    - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
    - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
    image: k8s.gcr.io/kube-apiserver:v1.23.0
    imagePullPolicy: IfNotPresent
    livenessProbe:
```

1 2 3 4 5 6 7 8 9 10 11

Inspect the environment and identify the authorization modes configured on the cluster.

Check the `kube-apiserver` settings.

- Node

```
      path: /usr/share/ca-certificates
      type: DirectoryOrCreate
    name: usr-share-ca-certificates
status: {}

root@controlplane ~ ➜

root@controlplane ~ ➜

root@controlplane ~ ➜  ps -aux | grep authorization
root      3403  0.0  0.0 830588 115420 ?        Ssl  22:54   0:55 kube-controller-manager --allocate-node
-cidrs=true --authentication-kubeconfig=/etc/kubernetes/controller-manager.conf --authorization-kubeconf
ig=/etc/kubernetes/controller-manager.conf --bind-address=127.0.0.1 --client-ca-file=/etc/kubernetes/pki
/ca.crt --cluster-cidr=10.244.0.0/16 --cluster-name=kubernetes --cluster-signing-cert-file=/etc/kubernet
es/pki/ca.crt --cluster-signing-key-file=/etc/kubernetes/pki/ca.key --controllers=*,bootstrapsigner,toke
```

Check the `kube-apiserver` settings.

- Node
- RBAC
- ABAC
- Node,RBAC

```
root@controlplane ~ ➜  ps -aux | grep authorization
root      3403  0.0  0.0 830588 115420 ?        Ssl  22:54   0:55 kube-controller-manager --allocate-node
-cidrs=true --authentication-kubeconfig=/etc/kubernetes/controller-manager.conf --authorization-kubeconf
ig=/etc/kubernetes/controller-manager.conf --bind-address=127.0.0.1 --client-ca-file=/etc/kubernetes/pki
/ca.crt --cluster-cidr=10.244.0.0/16 --cluster-name=kubernetes --cluster-signing-cert-file=/etc/kubernet
es/pki/ca.crt --cluster-signing-key-file=/etc/kubernetes/pki/ca.key --controllers=*,bootstrapsigner,toke
ncleaner --kubeconfig=/etc/kubernetes/controller-manager.conf --leader-elect=true --requestheader-client
-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt --root-ca-file=/etc/kubernetes/pki/ca.crt --service-acco
unt-private-key-file=/etc/kubernetes/pki/sa.key --service-cluster-ip-range=10.96.0.0/12 --use-service-ac
count-credentials=true
root      3614  0.0  0.0 759136 55292 ?        Ssl  22:55   0:10 kube-scheduler --authentication-kubecon
fig=/etc/kubernetes/scheduler.conf --authorization-kubeconfig=/etc/kubernetes/scheduler.conf --bind-addr
ess=127.0.0.1 --kubeconfig=/etc/kubernetes/scheduler.conf --leader-elect=true
root      3630  0.0  0.1 1118196 316984 ?      Ssl  22:55   2:07 kube-apiserver --advertise-address=10.4
8.174.6 --allow-privileged=true --authorization-mode=Node,RBAC --client-ca-file=/etc/kubernetes/pki/ca.c
rt --enable-admission-plugins=NodeRestriction --enable-bootstrap-token-auth=true --etcd-cafile=/etc/kube
rnetes/pki/etcd/ca.crt --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt --etcd-keyfile=/etc
/kubernetes/pki/apiserver-etcd-client.key --etcd-servers=https://127.0.0.1:2379 --kubelet-client-certifi
cate=/etc/kubernetes/pki/apiserver-kubelet-client.crt --kubelet-client-key=/etc/kubernetes/pki/apiserver
-kubelet-client.key --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname --proxy-client-cert
-file=/etc/kubernetes/pki/front-proxy-client.crt --proxy-client-key-file=/etc/kubernetes/pki/front-proxy
-client.key --requestheader-allowed-names=front-proxy-client --requestheader-client-ca-file=/etc/kuberne
tes/pki/front-proxy-ca.crt --requestheader-extra-headers-prefix=X-Remote-Extra- --requestheader-group-he
aders=X-Remote-Group --requestheader-username-headers=X-Remote-User --secure-port=6443 --service-account
-issuer=https://kubernetes.default.svc.cluster.local --service-account-key-file=/etc/kubernetes/pki/sa.p
ub --service-account-signing-key-file=/etc/kubernetes/pki/sa.key --service-cluster-ip-range=10.96.0.0/12
 --tls-cert-file=/etc/kubernetes/pki/apiserver.crt --tls-private-key-file=/etc/kubernetes/pki/apiserver.
key
root     25283  0.0  0.0  13444  1068 pts/0    S+   23:40   0:00 grep --color=auto authorization

root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

How many roles exist in the `default` namespace?

- 0
- 1
- 2
- 3

```
root@controlplane ~ ➜

root@controlplane ~ ➜

root@controlplane ~ ➜  k get roles
No resources found in default namespace.

root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

How many roles exist in all namespaces together?

0

1

2

5

10

15

12

```
root@controlplane ~ ➜

root@controlplane ~ ➜

root@controlplane ~ ➜  k get roles
No resources found in default namespace.

root@controlplane ~ ➜  k get roles -A
NAMESPACE     NAME                                                   CREATED AT
blue          developer                                              2022-04-17T23:25:04Z
kube-public   kubeadm:bootstrap-signer-clusterinfo                   2022-04-17T22:55:12Z
kube-public   system:controller:bootstrap-signer                     2022-04-17T22:55:10Z
kube-system   extension-apiserver-authentication-reader              2022-04-17T22:55:10Z
kube-system   kube-proxy                                             2022-04-17T22:55:12Z
kube-system   kubeadm:kubelet-config-1.23                            2022-04-17T22:55:11Z
kube-system   kubeadm:nodes-kubeadm-config                           2022-04-17T22:55:11Z
kube-system   system::leader-locking-kube-controller-manager          2022-04-17T22:55:10Z
kube-system   system::leader-locking-kube-scheduler                  2022-04-17T22:55:10Z
kube-system   system:controller:bootstrap-signer                     2022-04-17T22:55:10Z
kube-system   system:controller:cloud-provider                       2022-04-17T22:55:10Z
kube-system   system:controller:token-cleaner                        2022-04-17T22:55:10Z

root@controlplane ~ ➜  k get roles -A --no-headers
blue          developer                                              2022-04-17T23:25:04Z
kube-public   kubeadm:bootstrap-signer-clusterinfo                   2022-04-17T22:55:12Z
kube-public   system:controller:bootstrap-signer                     2022-04-17T22:55:10Z
kube-system   extension-apiserver-authentication-reader              2022-04-17T22:55:10Z
kube-system   kube-proxy                                             2022-04-17T22:55:12Z
kube-system   kubeadm:kubelet-config-1.23                            2022-04-17T22:55:11Z
kube-system   kubeadm:nodes-kubeadm-config                           2022-04-17T22:55:11Z
kube-system   system::leader-locking-kube-controller-manager          2022-04-17T22:55:10Z
kube-system   system::leader-locking-kube-scheduler                  2022-04-17T22:55:10Z
kube-system   system:controller:bootstrap-signer                     2022-04-17T22:55:10Z
kube-system   system:controller:cloud-provider                       2022-04-17T22:55:10Z
kube-system   system:controller:token-cleaner                        2022-04-17T22:55:10Z

root@controlplane ~ ➜  k get roles -A --no-headers | wc -l
12
```

1 2 3 4 5 6 7 8 9 10 11

What are the resources the `kube-proxy` role in the `kube-system` namespace is given access to?

pods

deployments

configmaps

secrets


```
blue          developer                                              2022-04-17T23:25:04Z
kube-public   kubeadm:bootstrap-signer-clusterinfo                   2022-04-17T22:55:12Z
kube-public   system:controller:bootstrap-signer                     2022-04-17T22:55:10Z
kube-system   extension-apiserver-authentication-reader              2022-04-17T22:55:10Z
kube-system   kube-proxy                                             2022-04-17T22:55:12Z
kube-system   kubeadm:kubelet-config-1.23                            2022-04-17T22:55:11Z
kube-system   kubeadm:nodes-kubeadm-config                           2022-04-17T22:55:11Z
kube-system   system::leader-locking-kube-controller-manager          2022-04-17T22:55:10Z
kube-system   system::leader-locking-kube-scheduler                  2022-04-17T22:55:10Z
kube-system   system:controller:bootstrap-signer                     2022-04-17T22:55:10Z
kube-system   system:controller:cloud-provider                       2022-04-17T22:55:10Z
kube-system   system:controller:token-cleaner                        2022-04-17T22:55:10Z

root@controlplane ~ ➜  k get roles -A --no-headers
blue          developer                                              2022-04-17T23:25:04Z
kube-public   kubeadm:bootstrap-signer-clusterinfo                   2022-04-17T22:55:12Z
kube-public   system:controller:bootstrap-signer                     2022-04-17T22:55:10Z
kube-system   extension-apiserver-authentication-reader              2022-04-17T22:55:10Z
kube-system   kube-proxy                                             2022-04-17T22:55:12Z
kube-system   kubeadm:kubelet-config-1.23                            2022-04-17T22:55:11Z
kube-system   kubeadm:nodes-kubeadm-config                           2022-04-17T22:55:11Z
kube-system   system::leader-locking-kube-controller-manager          2022-04-17T22:55:10Z
kube-system   system::leader-locking-kube-scheduler                  2022-04-17T22:55:10Z
kube-system   system:controller:bootstrap-signer                     2022-04-17T22:55:10Z
kube-system   system:controller:cloud-provider                       2022-04-17T22:55:10Z
kube-system   system:controller:token-cleaner                        2022-04-17T22:55:10Z

root@controlplane ~ ➜  k get roles -A --no-headers | wc -l
12
```


```
root@controlplane ~ ➜  k describe role kube-proxy -n kube-system
Name:         kube-proxy
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources   Non-Resource URLs  Resource Names  Verbs
  ---------   -----------------  --------------  -----
  configmaps  []                 [kube-proxy]    [get]

root@controlplane ~ ➜  []
```

1 2 3 4 5 6 7 8 9 10 11

What actions can the `kube-proxy` role perform on `configmaps` ?

Delete

List

Get

Update

```
blue          developer                                          2022-04-17T23:25:04Z
kube-public   kubeadm:bootstrap-signer-clusterinfo               2022-04-17T22:55:12Z
kube-public   system:controller:bootstrap-signer                 2022-04-17T22:55:10Z
kube-system   extension-apiserver-authentication-reader          2022-04-17T22:55:10Z
kube-system   kube-proxy                                         2022-04-17T22:55:12Z
kube-system   kubeadm:kubelet-config-1.23                        2022-04-17T22:55:11Z
kube-system   kubeadm:nodes-kubeadm-config                       2022-04-17T22:55:11Z
kube-system   system::leader-locking-kube-controller-manager     2022-04-17T22:55:10Z
kube-system   system::leader-locking-kube-scheduler              2022-04-17T22:55:10Z
kube-system   system:controller:bootstrap-signer                 2022-04-17T22:55:10Z
kube-system   system:controller:cloud-provider                   2022-04-17T22:55:10Z
kube-system   system:controller:token-cleaner                    2022-04-17T22:55:10Z

root@controlplane ~ ➜  k get roles -A --no-headers
blue          developer                                          2022-04-17T23:25:04Z
kube-public   kubeadm:bootstrap-signer-clusterinfo               2022-04-17T22:55:12Z
kube-public   system:controller:bootstrap-signer                 2022-04-17T22:55:10Z
kube-system   extension-apiserver-authentication-reader          2022-04-17T22:55:10Z
kube-system   kube-proxy                                         2022-04-17T22:55:12Z
kube-system   kubeadm:kubelet-config-1.23                        2022-04-17T22:55:11Z
kube-system   kubeadm:nodes-kubeadm-config                       2022-04-17T22:55:11Z
kube-system   system::leader-locking-kube-controller-manager     2022-04-17T22:55:10Z
kube-system   system::leader-locking-kube-scheduler              2022-04-17T22:55:10Z
kube-system   system:controller:bootstrap-signer                 2022-04-17T22:55:10Z
kube-system   system:controller:cloud-provider                   2022-04-17T22:55:10Z
kube-system   system:controller:token-cleaner                    2022-04-17T22:55:10Z

root@controlplane ~ ➜  k get roles -A --no-headers | wc -l
12

root@controlplane ~ ➜  k describe role kube-proxy -n kube-system
Name:         kube-proxy
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources   Non-Resource URLs  Resource Names  Verbs
  ---------   -----------------  --------------  -----
  configmaps  []                 [kube-proxy]    [get]

root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

Which of the following statements are true?

kube-proxy role can get details of configmap object by the name kube-proxy

kube-proxy role can delete the configmap it created

kube-proxy role can only view and update configmap object by the name kube-proxy


```
blue          developer                                          2022-04-17T23:25:04Z
kube-public   kubeadm:bootstrap-signer-clusterinfo               2022-04-17T22:55:12Z
kube-public   system:controller:bootstrap-signer                 2022-04-17T22:55:10Z
kube-system   extension-apiserver-authentication-reader          2022-04-17T22:55:10Z
kube-system   kube-proxy                                         2022-04-17T22:55:12Z
kube-system   kubeadm:kubelet-config-1.23                        2022-04-17T22:55:11Z
kube-system   kubeadm:nodes-kubeadm-config                       2022-04-17T22:55:11Z
kube-system   system::leader-locking-kube-controller-manager     2022-04-17T22:55:10Z
kube-system   system::leader-locking-kube-scheduler              2022-04-17T22:55:10Z
kube-system   system:controller:bootstrap-signer                 2022-04-17T22:55:10Z
kube-system   system:controller:cloud-provider                   2022-04-17T22:55:10Z
kube-system   system:controller:token-cleaner                    2022-04-17T22:55:10Z

root@controlplane ~ ➜  k get roles -A --no-headers
blue          developer                                          2022-04-17T23:25:04Z
kube-public   kubeadm:bootstrap-signer-clusterinfo               2022-04-17T22:55:12Z
kube-public   system:controller:bootstrap-signer                 2022-04-17T22:55:10Z
kube-system   extension-apiserver-authentication-reader          2022-04-17T22:55:10Z
kube-system   kube-proxy                                         2022-04-17T22:55:12Z
kube-system   kubeadm:kubelet-config-1.23                        2022-04-17T22:55:11Z
kube-system   kubeadm:nodes-kubeadm-config                       2022-04-17T22:55:11Z
kube-system   system::leader-locking-kube-controller-manager     2022-04-17T22:55:10Z
kube-system   system::leader-locking-kube-scheduler              2022-04-17T22:55:10Z
kube-system   system:controller:bootstrap-signer                 2022-04-17T22:55:10Z
kube-system   system:controller:cloud-provider                   2022-04-17T22:55:10Z
kube-system   system:controller:token-cleaner                    2022-04-17T22:55:10Z

root@controlplane ~ ➜  k get roles -A --no-headers | wc -l
12

root@controlplane ~ ➜  k describe role kube-proxy -n kube-system
Name:         kube-proxy
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources   Non-Resource URLs  Resource Names  Verbs
  ---------   -----------------  --------------  -----
  configmaps  []                 [kube-proxy]    [get]

root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

Which account is the `kube-proxy` role assigned to it?

- admin user
- User: kube-proxy
- kube-system
- ServiceAccount: kube-proxy
- Group: system:bootstrappers:kubeadm:default-node-token

```
root@controlplane ~ ➜

root@controlplane ~ ➜  k get rolebindings -n kube-system
NAME                                                   ROLE
  AGE
kube-proxy                                             Role/kube-proxy
  48m
kubeadm:kubelet-config-1.23                            Role/kubeadm:kubelet-config-1.23
  48m
kubeadm:nodes-kubeadm-config                           Role/kubeadm:nodes-kubeadm-config
  48m
system::extension-apiserver-authentication-reader      Role/extension-apiserver-authentication-reader
  48m
system::leader-locking-kube-controller-manager         Role/system::leader-locking-kube-controller-manager
  48m
system::leader-locking-kube-scheduler                  Role/system::leader-locking-kube-scheduler
  48m
system:controller:bootstrap-signer                     Role/system:controller:bootstrap-signer
  48m
system:controller:cloud-provider                       Role/system:controller:cloud-provider
  48m
system:controller:token-cleaner                        Role/system:controller:token-cleaner
  48m

root@controlplane ~ ➜  k describe rolebindings kube-proxy
Error from server (NotFound): rolebindings.rbac.authorization.k8s.io "kube-proxy" not found

root@controlplane ~ ✖ k describe rolebindings kube-proxy -n kube-system
Name:         kube-proxy
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  kube-proxy
Subjects:
  Kind   Name                                           Namespace
  ----   ----                                           ---------
  Group  system:bootstrappers:kubeadm:default-node-token

root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

A user `dev-user` is created. User's details have been added to the `kubeconfig` file. Inspect the permissions granted to the user. Check if the user can list pods in the `default` namespace.

Use the `--as dev-user` option with `kubectl` to run commands as the `dev-user`.

dev-user has permissions to list pods

dev-user does not have permissions to list pods

```
root@controlplane ~ ➜  kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://controlplane:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: dev-user
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
- name: kubernetes-admin
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED

root@controlplane ~ ➜  k get pods --as dev-user
Error from server (Forbidden): pods is forbidden: User "dev-user" cannot list resource "pods" in API gro
up "" in the namespace "default"

root@controlplane ~ ✖
```

1 2 3 4 5 6 7 8 9 10 11

Create the necessary roles and role bindings required for the `dev-user` to create, list and delete pods in the `default` namespace. Use the given spec:

Check

- Role: developer
- Role Resources: pods
- Role Actions: list
- Role Actions: create
- Role Actions: delete
- RoleBinding: dev-user-binding
- RoleBinding: Bound to dev-user

```
root@controlplane ~ ➜  k create role --help
Create a role with single rule.

Examples:
  # Create a role named "pod-reader" that allows user to perform "get", "watch" and "list" on pods
  kubectl create role pod-reader --verb=get --verb=list --verb=watch --resource=pods

  # Create a role named "pod-reader" with ResourceName specified
  kubectl create role pod-reader --verb=get --resource=pods --resource-name=readablepod
--resource-name=anotherpod

  # Create a role named "foo" with API Group specified
  kubectl create role foo --verb=get,list,watch --resource=rs.extensions

  # Create a role named "foo" with SubResource specified
  kubectl create role foo --verb=get,list,watch --resource=pods,pods/status

Options:
      --allow-missing-template-keys=true: If true, ignore any errors in templates when a field or
map key is missing in the template. Only applies to golang and jsonpath output formats.
      --dry-run='none': Must be "none", "server", or "client". If client strategy, only print the
object that would be sent, without sending it. If server strategy, submit server-side request
without persisting the resource.
      --field-manager='kubectl-create': Name of the manager used to track field ownership.
  -o, --output='': Output format. One of:
json|yaml|name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-fil
e.
      --resource=[]: Resource that the rule applies to
      --resource-name=[]: Resource in the white list that the rule applies to, repeat this flag for
multiple items
      --save-config=false: If true, the configuration of current object will be saved in its
annotation. Otherwise, the annotation will be unchanged. This flag is useful when you want to
perform kubectl apply on this object in the future.
      --show-managed-fields=false: If true, keep the managedFields when printing objects in JSON or
YAML format.
      --template='': Template string or path to template file to use when -o=go-template,
-o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
      --validate=true: If true, use a schema to validate the input before sending it
```

1 2 3 4 5 6 7 8 9 10 11

Create the necessary roles and role bindings required for the `dev-user` to create, list and delete pods in the `default` namespace. Use the given spec:

Check

- Role: developer
- Role Resources: pods
- Role Actions: list
- Role Actions: create
- Role Actions: delete
- RoleBinding: dev-user-binding
- RoleBinding: Bound to dev-user

```
root@controlplane ~ ➜

root@controlplane ~ ➜  k create role developer --verb=list,create,delete --resources=pods
Error: unknown flag: --resources
See 'kubectl create role --help' for usage.

root@controlplane ~ ✖ k create role developer --verb=list,create,delete --resource=pods
role.rbac.authorization.k8s.io/developer created

root@controlplane ~ ➜

root@controlplane ~ ➜  k describe role developer
Name:         developer
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  pods       []                 []              [list create delete]

root@controlplane ~ ➜

root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

Create the necessary roles and role bindings required for the `dev-user` to create, list and delete pods in the `default` namespace.
Use the given spec:

Check

- Role: developer
- Role Resources: pods
- Role Actions: list
- Role Actions: create
- Role Actions: delete
- RoleBinding: dev-user-binding
- RoleBinding: Bound to dev-user

```
root@controlplane ~ ➜  k create rolebinding --help
Create a role binding for a particular role or cluster role.

Examples:
  # Create a role binding for user1, user2, and group1 using the admin cluster role
  kubectl create rolebinding admin --clusterrole=admin --user=user1 --user=user2 --group=group1

Options:
      --allow-missing-template-keys=true: If true, ignore any errors in templates when a field or
map key is missing in the template. Only applies to golang and jsonpath output formats.
      --clusterrole='': ClusterRole this RoleBinding should reference
      --dry-run='none': Must be "none", "server", or "client". If client strategy, only print the
object that would be sent, without sending it. If server strategy, submit server-side request
without persisting the resource.
      --field-manager='kubectl-create': Name of the manager used to track field ownership.
      --group=[]: Groups to bind to the role
  -o, --output='': Output format. One of:
json|yaml|name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-fil
e.
      --role='': Role this RoleBinding should reference
      --save-config=false: If true, the configuration of current object will be saved in its
annotation. Otherwise, the annotation will be unchanged. This flag is useful when you want to
perform kubectl apply on this object in the future.
      --serviceaccount=[]: Service accounts to bind to the role, in the format <namespace>:<name>
      --show-managed-fields=false: If true, keep the managedFields when printing objects in JSON or
YAML format.
      --template='': Template string or path to template file to use when -o=go-template,
-o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
      --validate=true: If true, use a schema to validate the input before sending it

Usage:
  kubectl create rolebinding NAME --clusterrole=NAME|--role=NAME [--user=username]
[--group=groupname] [--serviceaccount=namespace:serviceaccountname] [--dry-run=server|client|none]
[options]

Use "kubectl options" for a list of global command-line options (applies to all commands).

root@controlplane ~ ➜  k create rolebinding dev-user-binding --role=developer --user=dev-user
```

```
root@controlplane ~ ➜  k create rolebinding dev-user-binding --role=developer --user=dev-user
rolebinding.rbac.authorization.k8s.io/dev-user-binding created

root@controlplane ~ ➜

root@controlplane ~ ➜  k describe rolebinding dev-user-binding
Name:         dev-user-binding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  developer
Subjects:
  Kind  Name      Namespace
  ----  ----      ---------
  User  dev-user

root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

The `dev-user` is trying to get details about the `dark-blue-app` pod in the `blue` namespace.
Investigate and fix the issue.
We have created the required roles and rolebindings, but something seems to be wrong.

Check

- Issue Fixed

```
root@controlplane ~ ➜  k --as dev-user get pod dark-blue-app -blue
Error: unknown shorthand flag: 'b' in -blue
See 'kubectl get --help' for usage.

root@controlplane ~ ✖ k --as dev-user get pod dark-blue-app -n blue
Error from server (Forbidden): pods "dark-blue-app" is forbidden: User "dev-user" cannot get resource "p
ods" in API group "" in the namespace "blue"

root@controlplane ~ ✖

root@controlplane ~ ✖ k get roles -n blue
NAME        CREATED AT
developer   2022-04-17T23:25:04Z

root@controlplane ~ ➜  k get rolebindings -n blue
NAME                ROLE              AGE
dev-user-binding    Role/developer    24m

root@controlplane ~ ➜

root@controlplane ~ ➜  k describe role developer -n blue
Name:         developer
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  pods       []                 [blue-app]      [get watch create delete]

root@controlplane ~ ➜  k edit role developer -n blue
```

1 2 3 4 5 6 7 8 9 10 11

The `dev-user` is trying to get details about the `dark-blue-app` pod in the `blue` namespace.

Investigate and fix the issue.
We have created the required roles and rolebindings, but something seems to be wrong.

Check

- Issue Fixed

```
root@controlplane ~ ➜  k --as dev-user get pod dark-blue-app -blue
Error: unknown shorthand flag: 'b' in -blue
See 'kubectl get --help' for usage.

root@controlplane ~ ✖ k --as dev-user get pod dark-blue-app -n blue
Error from server (Forbidden): pods "dark-blue-app" is forbidden: User "dev-user" cannot get resource "p
ods" in API group "" in the namespace "blue"

root@controlplane ~ ✖

root@controlplane ~ ✖ k get roles -n blue
NAME        CREATED AT
developer   2022-04-17T23:25:04Z

root@controlplane ~ ➜  k get rolebindings -n blue
NAME               ROLE             AGE
dev-user-binding   Role/developer   24m

root@controlplane ~ ➜

root@controlplane ~ ➜  k describe role developer -n blue
Name:         developer
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  pods       []                 [blue-app]      [get watch create delete]

root@controlplane ~ ➜  k edit role developer -n blue
role.rbac.authorization.k8s.io/developer edited

root@controlplane ~ ➜  k --as dev-user get pod dark-blue-app -n blue
NAME            READY   STATUS    RESTARTS   AGE
dark-blue-app   1/1     Running   0          25m

root@controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

Grant the `dev-user` permissions to create deployments in the `blue` namespace.

Remember to add for api group `"apps"`.

Check

- Create Deployments

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  creationTimestamp: "2022-04-17T23:25:04Z"
  name: developer
  namespace: blue
  resourceVersion: "4534"
  uid: b1424ab8-9776-43b0-bddf-433dfff73c1e
rules:
- apiGroups:
  - ""
  resourceNames:
  - dark-blue-app
  resources:
  - pods
  verbs:
  - get
  - watch
  - create
  - delete
- apiGroups:
  - "apps"
  resources:
  - deployment
  verbs:
  - get
  - watch
  - create
  - delete
~
~
~
~
~
~
```

1 2 3 4 5 6 7 8 9 10 11

Grant the `dev-user` permissions to create deployments in the `blue` namespace. Remember to add for api group `"apps"`.

Check

- Create Deployments

```
root@controlplane ~ ➜  k --as dev-user create deployment nginx --image=nginx -n blue
error: failed to create deployment: deployments.apps is forbidden: User "dev-user" cannot create resourc
e "deployments" in API group "apps" in the namespace "blue"

root@controlplane ~ ✖ k edit role developer -n blue
role.rbac.authorization.k8s.io/developer edited

root@controlplane ~ ➜

root@controlplane ~ ➜  k describe role dev
```

1 2 3 4 5 6 7 8 9 10 11

Grant the `dev-user` permissions to create deployments in the `blue` namespace. Remember to add for api group `"apps"`.

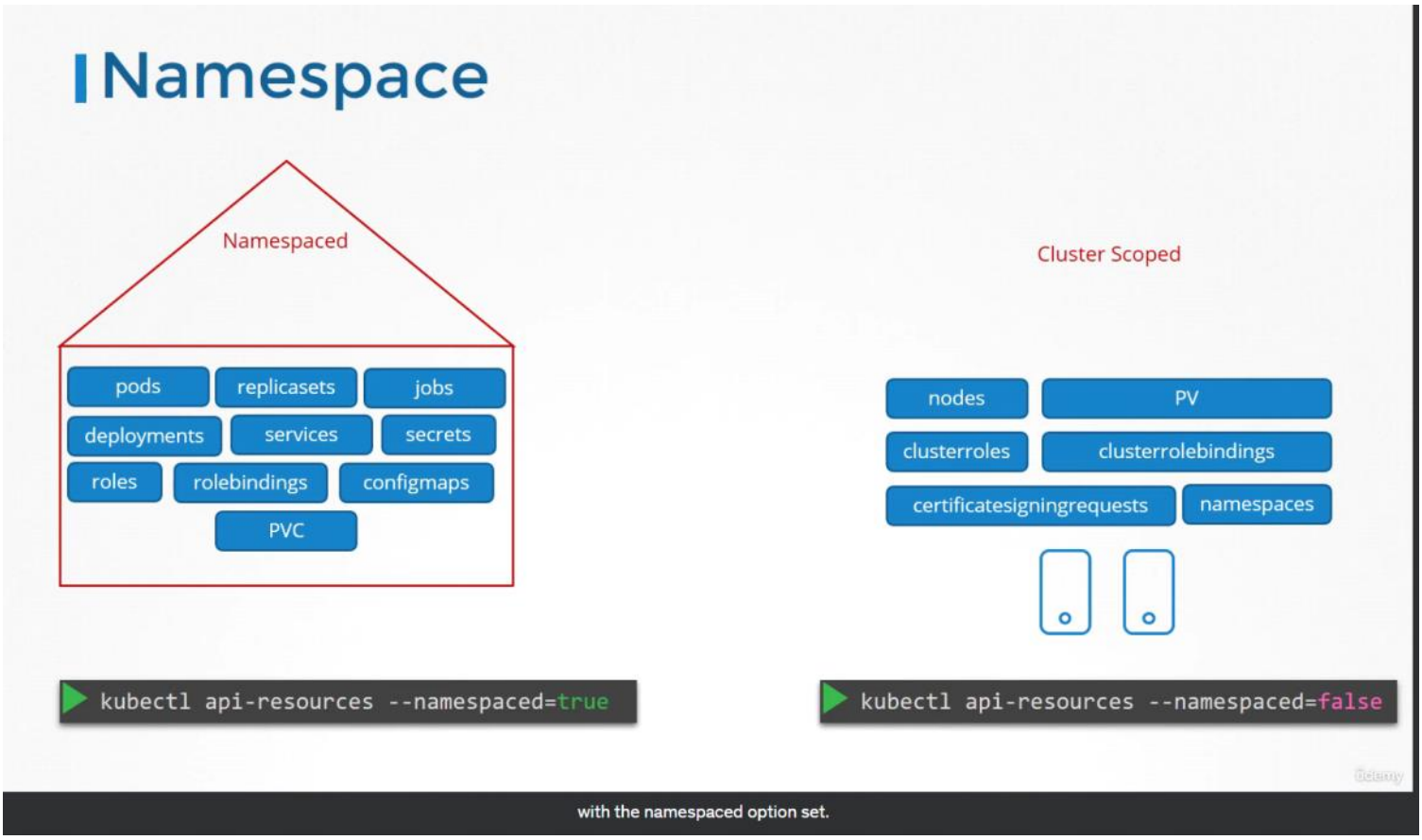
Check

- Create Deployments

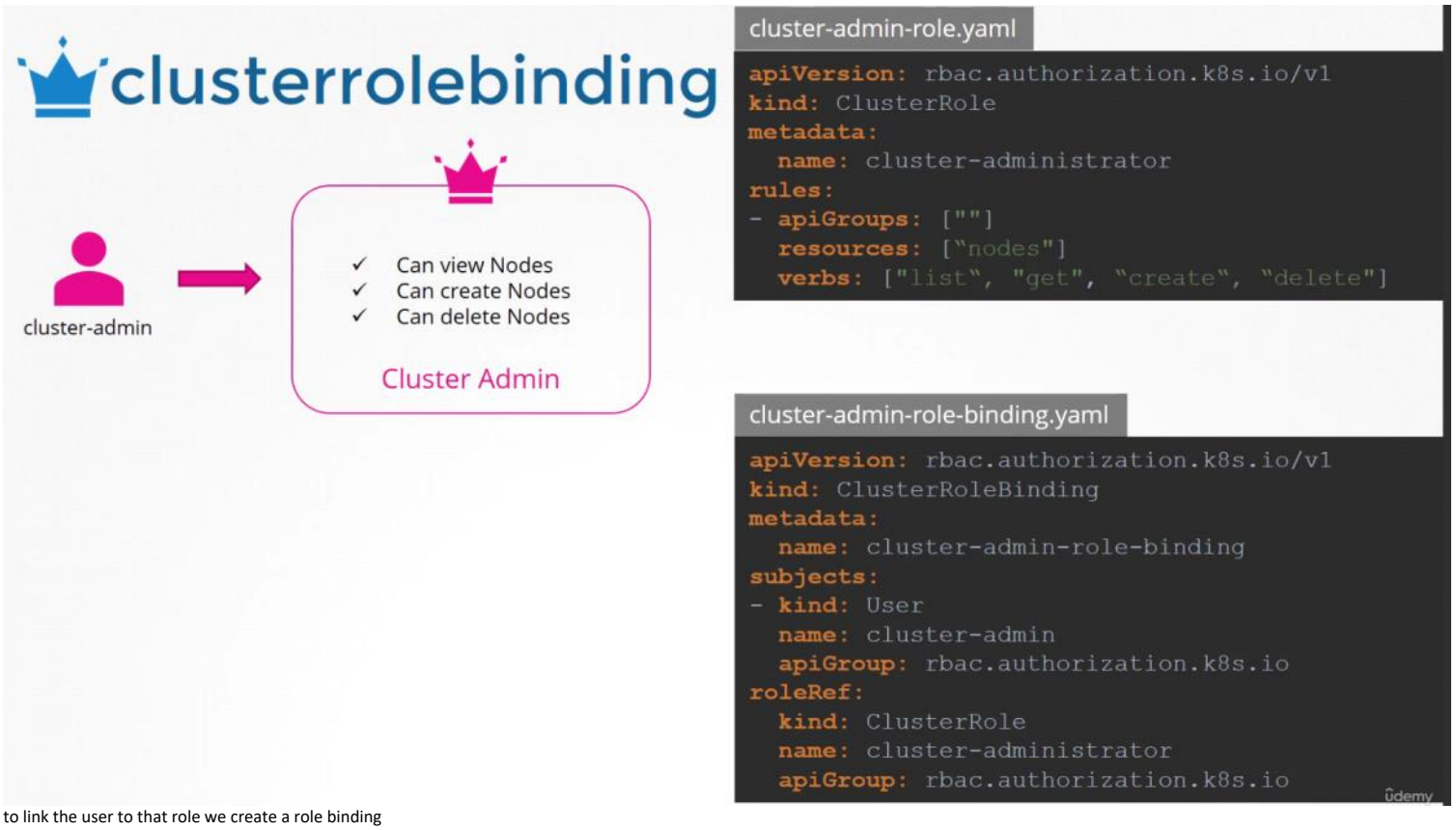
```
root@controlplane ~ ➜  k --as dev-user create deployment nginx --image=nginx -n blue
error: failed to create deployment: deployments.apps is forbidden: User "dev-user" cannot create resourc
e "deployments" in API group "apps" in the namespace "blue"

root@controlplane ~ ✖ k edit role developer -n blue
role.rbac.authorization.k8s.io/developer edited

root@controlplane ~ ➜

root@controlplane ~ ➜  k describe role developer -n blue
Name:         developer
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources        Non-Resource URLs  Resource Names   Verbs
  ---------        -----------------  --------------   -----
  pods             []                 [dark-blue-app]  [get watch create delete]
  deployment.apps  []                 []               [get watch create delete]

root@controlplane ~ ➜  k --as dev-user create deployment nginx --image=nginx -n blue
error: failed to create deployment: deployments.apps is forbidden: User "dev-user" cannot create resourc
e "deployments" in API group "apps" in the namespace "blue"

root@controlplane ~ ✖ k edit role developer -n blue
role.rbac.authorization.k8s.io/developer edited

root@controlplane ~ ➜  k --as dev-user create deployment nginx --image=nginx -n blue
deployment.apps/nginx created

root@controlplane ~ ➜

root@controlplane ~ ➜
```

Cluster role and role bindings

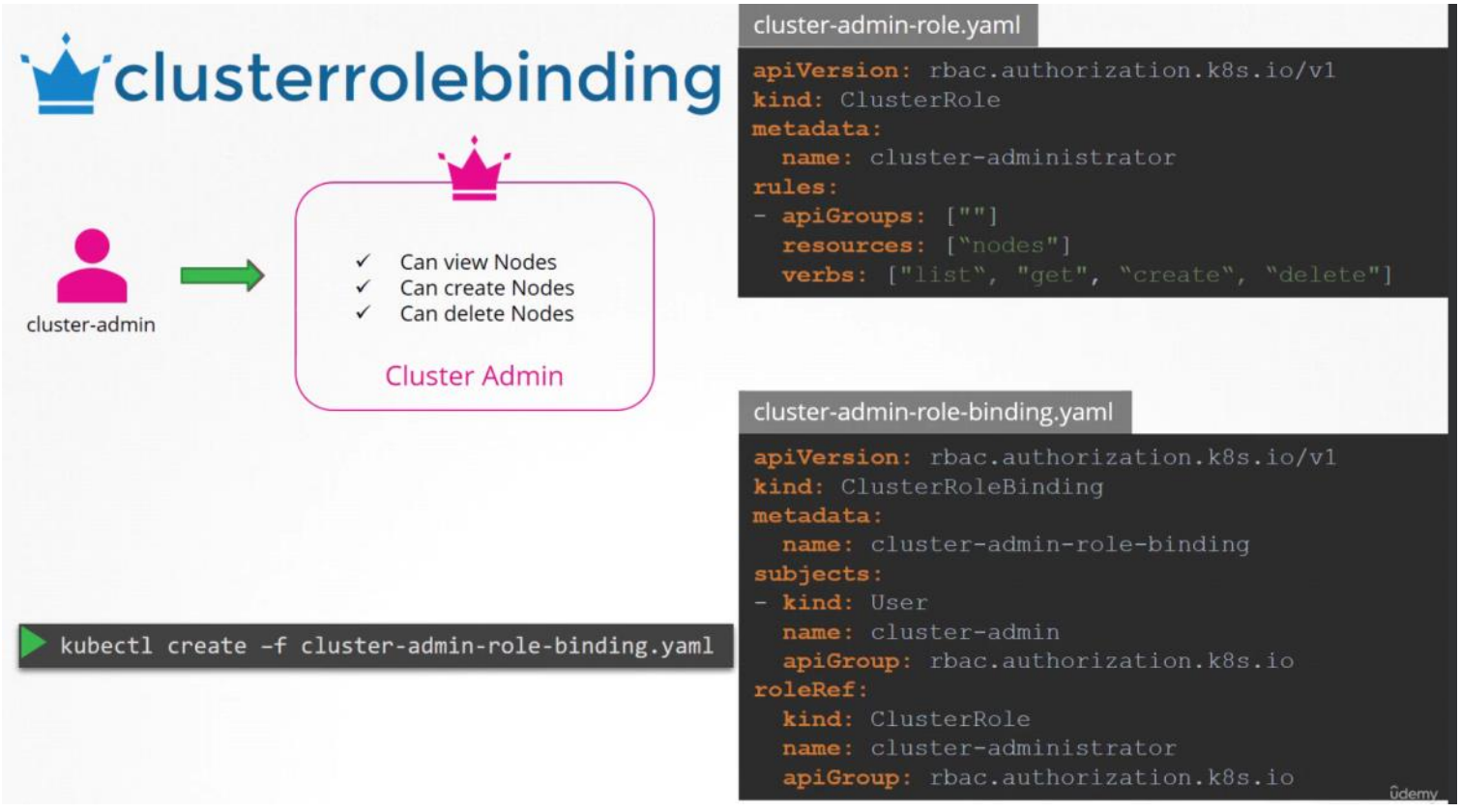
Tuesday, July 2, 2024 9:42 PM

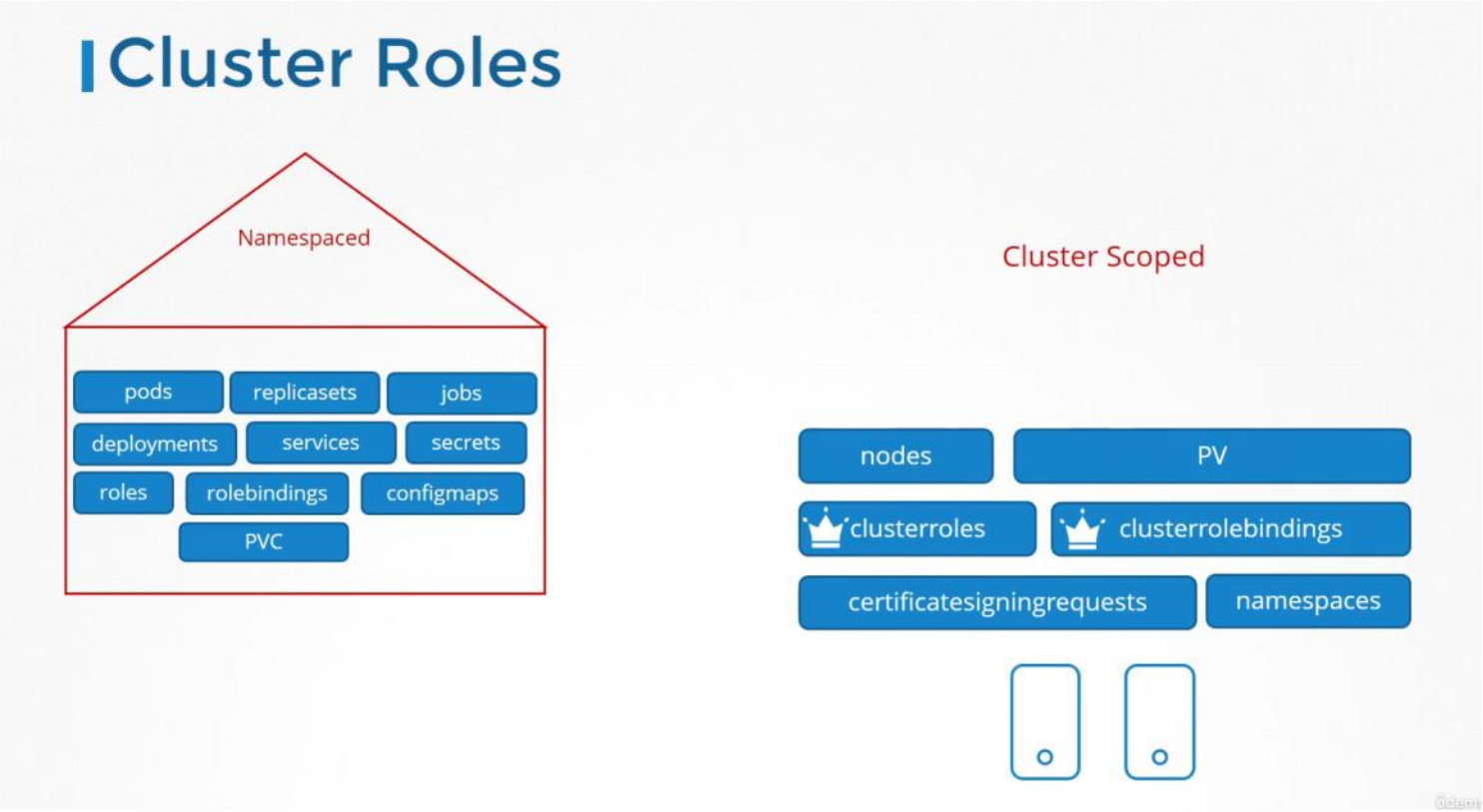


roles made rules astat jyat resources ani verbs define astat.



to link the user to that role we create a role binding




Cluster Roles
Namespaced
pods
replicasets
jobs
deployments
services
secrets
roles
rolebindings
configmaps
PVC
Cluster Scoped
nodes
PV
clusterroles
clusterrolebindings
certificatesigningrequests
namespaces

Practice- Cluster role and cluster role bindings

Tuesday, July 2, 2024 9:50 PM

1 2 3 4 5 6 7 8

How many `ClusterRoles` do you see defined in the cluster?

0

10

34

119

69

```
system:controller:resourcequota-controller                    2022-04-18T00:01:23Z
system:controller:route-controller                            2022-04-18T00:01:23Z
system:controller:service-account-controller                  2022-04-18T00:01:23Z
system:controller:service-controller                          2022-04-18T00:01:23Z
system:controller:statefulset-controller                      2022-04-18T00:01:23Z
system:controller:ttl-controller                              2022-04-18T00:01:23Z
system:controller:certificate-controller                      2022-04-18T00:01:23Z
system:controller:pvc-protection-controller                   2022-04-18T00:01:23Z
system:controller:pv-protection-controller                    2022-04-18T00:01:23Z
system:controller:ttl-after-finished-controller               2022-04-18T00:01:23Z
system:controller:root-ca-cert-publisher                      2022-04-18T00:01:23Z
k3s-cloud-controller-manager                                  2022-04-18T00:01:27Z
system:coredns                                                2022-04-18T00:01:28Z
local-path-provisioner-role                                   2022-04-18T00:01:28Z
system:aggregated-metrics-reader                              2022-04-18T00:01:28Z
system:metrics-server                                         2022-04-18T00:01:29Z
system:k3s-controller                                         2022-04-18T00:01:29Z
view                                                          2022-04-18T00:01:23Z
edit                                                          2022-04-18T00:01:23Z
admin                                                         2022-04-18T00:01:23Z
traefik                                                       2022-04-18T00:02:54Z

controlplane ~ ➜  k get clusterroles --no-headers | wc -k
wc: unrecognized option: k
BusyBox v1.34.1 (2021-11-23 00:57:35 UTC) multi-call binary.

Usage: wc [-cmlwL] [FILE]...

Count lines, words, and bytes for FILEs (or stdin)

        -c      Count bytes
        -m      Count characters
        -l      Count newlines
        -w      Count words
        -L      Print longest line length

controlplane ~ ✖ k get clusterroles --no-headers | wc -l
69

controlplane ~ ➜
```

1 2 3 4 5 6 7 8

How many `ClusterRoleBindings` exist on the cluster?

0

19

43

119

```
controlplane ~ ➜  k get clusterrolebindings --no-headers | wc -l
54

controlplane ~ ➜
```

1 2 3 4 5 6 7 8

What namespace is the `cluster-admin` clusterrole part of?

- kube-system
- kube-public
- default
- Cluster Roles are cluster wide and not part of any namespace

```
controlplane ~ ➜  k get clusterrolebindings --no-headers | wc -l
54

controlplane ~ ➜  k get clusterroles
NAME                                                                   CREATED AT
cluster-admin                                                          2022-04-18T00:01:23Z
system:discovery                                                       2022-04-18T00:01:23Z
system:monitoring                                                      2022-04-18T00:01:23Z
system:basic-user                                                      2022-04-18T00:01:23Z
system:public-info-viewer                                              2022-04-18T00:01:23Z
system:aggregate-to-admin                                              2022-04-18T00:01:23Z
system:aggregate-to-edit                                               2022-04-18T00:01:23Z
system:aggregate-to-view                                               2022-04-18T00:01:23Z
system:heapster                                                        2022-04-18T00:01:23Z
system:node                                                            2022-04-18T00:01:23Z
system:node-problem-detector                                           2022-04-18T00:01:23Z
system:kubelet-api-admin                                               2022-04-18T00:01:23Z
system:node-bootstrapper                                               2022-04-18T00:01:23Z
system:auth-delegator                                                  2022-04-18T00:01:23Z
system:kube-aggregator                                                 2022-04-18T00:01:23Z
system:kube-controller-manager                                         2022-04-18T00:01:23Z
system:kube-dns                                                        2022-04-18T00:01:23Z
system:persistent-volume-provisioner                                   2022-04-18T00:01:23Z
system:certificates.k8s.io:certificatesigningrequests:nodeclient       2022-04-18T00:01:23Z
system:certificates.k8s.io:certificatesigningrequests:selfnodeclient   2022-04-18T00:01:23Z
system:volume-scheduler                                                2022-04-18T00:01:23Z
system:certificates.k8s.io:legacy-unknown-approver                     2022-04-18T00:01:23Z
system:certificates.k8s.io:kubelet-serving-approver                    2022-04-18T00:01:23Z
system:certificates.k8s.io:kube-apiserver-client-approver              2022-04-18T00:01:23Z
system:certificates.k8s.io:kube-apiserver-client-kubelet-approver      2022-04-18T00:01:23Z
system:service-account-issuer-discovery                                2022-04-18T00:01:23Z
system:node-proxier                                                    2022-04-18T00:01:23Z
system:kube-scheduler                                                  2022-04-18T00:01:23Z
system:controller:attachdetach-controller                              2022-04-18T00:01:23Z
system:controller:clusterrole-aggregation-controller                   2022-04-18T00:01:23Z
system:controller:cronjob-controller                                   2022-04-18T00:01:23Z
system:controller:daemon-set-controller                                2022-04-18T00:01:23Z
system:controller:deployment-controller                                2022-04-18T00:01:23Z
system:controller:disruption-controller                                2022-04-18T00:01:23Z
```

1 2 3 4 5 6 7 8

What user/groups are the `cluster-admin` role bound to?
The ClusterRoleBinding for the role is with the same name.

- kube-admin
- system:masters
- system:nodes
- cluster-admin

```
controlplane ~ ➜  k get clusterrolebindings | grep cluster-admin
cluster-admin                                          ClusterRole/cluster-admin
                   12m
helm-kube-system-traefik                               ClusterRole/cluster-admin
                   12m
helm-kube-system-traefik-crd                           ClusterRole/cluster-admin
                   12m

controlplane ~ ➜  k describe clusterrolebindings cluster-admin
Name:         cluster-admin
Labels:       kubernetes.io/bootstrapping=rbac-defaults
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
Role:
  Kind:  ClusterRole
  Name:  cluster-admin
Subjects:
  Kind   Name            Namespace
  ----   ----            ---------
  Group  system:masters

controlplane ~ ➜  k describe clusterrolebindings helm-kube-system-traefik
Name:         helm-kube-system-traefik
Labels:       objectset.rio.cattle.io/hash=c42f5b5dd9ee50718523a82c68d4392a7dec9fc4
Annotations:  objectset.rio.cattle.io/applied:
                H4sIAAAAAAAAA/4xRTY/TQAz9K8jnJNC0oWkkDtADSNwK4oI4ODNOYzIf0YwTBFX/O0ra7nalbXePM/Z7fh8HwJ5/
UIjsHVQQalQZDtL6wP9Q2LusK2PG/u24gAQ6dhoq2JohCoWdN/...
              objectset.rio.cattle.io/id: helm-controller
              objectset.rio.cattle.io/owner-gvk: helm.cattle.io/v1, Kind=HelmChart
              objectset.rio.cattle.io/owner-name: traefik
              objectset.rio.cattle.io/owner-namespace: kube-system
Role:
  Kind:  ClusterRole
  Name:  cluster-admin
Subjects:
  Kind            Name          Namespace
  ----            ----          ---------
  ServiceAccount  helm-traefik  kube-system

controlplane ~ ➜
```

1 2 3 4 5 6 7 8

What level of permission does the `cluster-admin` role grant?

Inspect the `cluster-admin` role's privileges.

- Node level tasks only
- Pod level tasks only
- Security related tasks only
- Perform any action on any resource in the cluster

```
controplane ~ ➜  k describe clusterrole cluster-admin
Name:         cluster-admin
Labels:       kubernetes.io/bootstrapping=rbac-defaults
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  *.*        []                 []              [*]
             [*]                []              [*]

controplane ~ ➜
```

1 2 3 4 5 6 7 8

A new user `michelle` joined the team. She will be focusing on the `nodes` in the cluster. Create the required `ClusterRoles` and `ClusterRoleBindings` so she gets access to the `nodes`.

Check

- Grant permission to access nodes

```
controplane ~ ➜

controplane ~ ➜

controplane ~ ➜  kubectl create clusterrole michelle-role --verb=get,list,watch --resource=nodes
clusterrole.rbac.authorization.k8s.io/michelle-role created

controplane ~ ➜

controplane ~ ➜  kubectl c
```

1 2 3 4 5 6 7 8

A new user `michelle` joined the team. She will be focusing on the `nodes` in the cluster. Create the required `ClusterRoles` and `ClusterRoleBindings` so she gets access to the `nodes`.

Check

- Grant permission to access nodes

```
Usage:
  kubectl create clusterrolebinding NAME --clusterrole=NAME [--user=username] [--group=groupname]
[--serviceaccount=namespace:serviceaccountname] [--dry-run=server|client|none] [options]

Use "kubectl options" for a list of global command-line options (applies to all commands).

controlplane ~ ➜

controlplane ~ ➜

controlplane ~ ➜  kubectl create clusterrolebinding michelle-role-binding --clusterrole=michelle-role --
user=michelle
clusterrolebinding.rbac.authorization.k8s.io/michelle-role-binding created

controlplane ~ ➜

controlplane ~ ➜  k describe clusterrole michelle-role
Name:         michelle-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  nodes      []                 []              [get list watch]

controlplane ~ ➜

controlplane ~ ➜  k describe clusterrolebinding michelle-role-binding
Name:         michelle-role-binding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  michelle-role
Subjects:
  Kind  Name      Namespace
  ----  ----      ---------
  User  michelle

controlplane ~ ➜
```

1 2 3 4 5 6 7 8

A new user `michelle` joined the team. She will be focusing on the `nodes` in the cluster. Create the required `ClusterRoles` and `ClusterRoleBindings` so she gets access to the `nodes`.

...

```
clusterrolebinding.rbac.authorization.k8s.io/michelle-role-binding created

controlplane ~ ➜

To exit full screen, press Esc

controlplane ~ ➜  k describe clusterrole michelle-role
Name:         michelle-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  nodes      []                 []              [get list watch]

controlplane ~ ➜

controlplane ~ ➜  k describe clusterrolebinding michelle-role-binding
Name:         michelle-role-binding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  michelle-role
Subjects:
  Kind  Name      Namespace
  ----  ----      ---------
  User  michelle

controlplane ~ ➜

controlplane ~ ➜

controlplane ~ ➜  k get nodes --as michelle
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   16m   v1.23.3+k3s1

controlplane ~ ➜

controlplane ~ ➜

controlplane ~ ➜
```

1 2 3 4 5 6 7 8

`michelle` 's responsibilities are growing and now she will be responsible for storage as well. Create the required `ClusterRoles` and `ClusterRoleBindings` to allow her access to Storage.
Get the API groups and resource names from command `kubectl api-resources`. Use the given spec:

Check

- ClusterRole: storage-admin
- Resource: persistentvolumes
- Resource: storageclasses
- ClusterRoleBinding: michelle-storage-admin
- ClusterRoleBinding Subject: michelle
- ClusterRoleBinding Role: storage-admin

```
ingressroutetcps                          traefik.containo.us/v1alpha1          true        Ingre
ssRouteTCP
ingressrouteudps                          traefik.containo.us/v1alpha1          true        Ingre
ssRouteUDP
middlewares                               traefik.containo.us/v1alpha1          true        Middl
eware
middlewaretcps                            traefik.containo.us/v1alpha1          true        Middl
ewareTCP
serverstransports                         traefik.containo.us/v1alpha1          true        Serve
rsTransport
tlsoptions                                traefik.containo.us/v1alpha1          true        TLSOp
tion
tlsstores                                 traefik.containo.us/v1alpha1          true        TLSSt
ore
traefikservices                           traefik.containo.us/v1alpha1          true        Traef
ikService

controlplane ~ ➜

controlplane ~ ➜  k create clusterrole storage-admin --resource=persistentvolumes,storageclasses --verb=
list,create,get,watch
clusterrole.rbac.authorization.k8s.io/storage-admin created

controlplane ~ ➜

controlplane ~ ➜

controlplane ~ ➜  k describe clusterrole storage-admin
Name:         storage-admin
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources                       Non-Resource URLs  Resource Names  Verbs
  ---------                       -----------------  --------------  -----
  persistentvolumes               []                 []              [list create get watch]
  storageclasses.storage.k8s.io   []                 []              [list create get watch]

controlplane ~ ➜

controlplane ~ ➜  k get clusterrole storage-admin
```

```
controlplane ~ ➜  kubectl api-resources
NAME                              SHORTNAMES   APIVERSION                             NAMESPACED   KIND
bindings                                       v1                                     true         Binding
componentstatuses                 cs           v1                                     false        ComponentStatus
configmaps                        cm           v1                                     true         ConfigMap
endpoints                         ep           v1                                     true         Endpoints
events                            ev           v1                                     true         Event
limitranges                       limits       v1                                     true         LimitRange
namespaces                        ns           v1                                     false        Namespace
nodes                             no           v1                                     false        Node
persistentvolumeclaims            pvc          v1                                     true         PersistentVolumeClaim
persistentvolumes                 pv           v1                                     false        PersistentVolume
pods                              po           v1                                     true         Pod
podtemplates                                   v1                                     true         PodTemplate
replicationcontrollers            rc           v1                                     true         ReplicationController
resourcequotas                    quota        v1                                     true         ResourceQuota
secrets                                        v1                                     true         Secret
serviceaccounts                   sa           v1                                     true         ServiceAccount
services                          svc          v1                                     true         Service
mutatingwebhookconfigurations                  admissionregistration.k8s.io/v1        false        MutatingWebhookConfiguration
validatingwebhookconfigurations                admissionregistration.k8s.io/v1        false        ValidatingWebhookConfiguration
customresourcedefinitions         crd,crds     apiextensions.k8s.io/v1                false        CustomResourceDefinition
apiservices                                    apiregistration.k8s.io/v1              false        APIService
controllerrevisions                            apps/v1                                true         ControllerRevision
daemonsets                        ds           apps/v1                                true         DaemonSet
deployments                       deploy       apps/v1                                true         Deployment
replicasets                       rs           apps/v1                                true         ReplicaSet
statefulsets                      sts          apps/v1                                true         StatefulSet
tokenreviews                                   authentication.k8s.io/v1               false        TokenReview
localsubjectaccessreviews                      authorization.k8s.io/v1                true         LocalSubjectAccessReview
selfsubjectaccessreviews                       authorization.k8s.io/v1                false        SelfSubjectAccessReview
selfsubjectrulesreviews                        authorization.k8s.io/v1                false        SelfSubjectRulesReview
subjectaccessreviews                           authorization.k8s.io/v1                false        SubjectAccessReview
horizontalpodautoscalers          hpa          autoscaling/v2                         true         HorizontalPodAutoscaler
cronjobs                          cj           batch/v1                               true         CronJob
jobs                                           batch/v1                               true         Job
certificatesigningrequests        csr          certificates.k8s.io/v1                 false        CertificateSigningRequest
leases                                         coordination.k8s.io/v1                 true         Lease
endpointslices                                 discovery.k8s.io/v1                    true         EndpointSlice
```

1 2 3 4 5 6 7 8

`michelle` 's responsibilities are growing and now she will be responsible for storage as well. Create the required `ClusterRoles` and `ClusterRoleBindings` to allow her access to Storage.
Get the API groups and resource names from command `kubectl api-resources`. Use the given spec:

Check

- ClusterRole: storage-admin
- Resource: persistentvolumes
- Resource: storageclasses
- ClusterRoleBinding: michelle-storage-admin
- ClusterRoleBinding Subject: michelle
- ClusterRoleBinding Role: storage-admin

```
Annotations:  <none>
PolicyRule:
  Resources                      Non-Resource URLs  Resource Names  Verbs
  ---------                      -----------------  --------------  -----
  persistentvolumes              []                 []              [list create get watch]
  storageclasses.storage.k8s.io  []                 []              [list create get watch]

controlplane ~ ➜

controlplane ~ ➜  k get clusterrole storage-admin -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  creationTimestamp: "2022-04-18T00:20:48Z"
  name: storage-admin
  resourceVersion: "921"
  uid: e0ee52a7-b32c-4a42-b8b7-a783b040cd4e
rules:
- apiGroups:
  - ""
  resources:
  - persistentvolumes
  verbs:
  - list
  - create
  - get
  - watch
- apiGroups:
  - storage.k8s.io
  resources:
  - storageclasses
  verbs:
  - list
  - create
  - get
  - watch

controlplane ~ ➜

controlplane ~ ➜
```

1 2 3 4 5 6 7 8

`michelle` 's responsibilities are growing and now she will be responsible for storage as well. Create the required `ClusterRoles` and `ClusterRoleBindings` to allow her access to Storage.
Get the API groups and resource names from command `kubectl api-resources`. Use the given spec:

Complete Next

- ✔ ClusterRole: storage-admin
- ✔ Resource: persistentvolumes
- ✔ Resource: storageclasses
- ✔ ClusterRoleBinding: michelle-storage-admin
- ✔ ClusterRoleBinding Subject: michelle
- ✔ ClusterRoleBinding Role: storage-admin

```
controlplane ~ ➜  k create clusterrolebinding michelle-storage-admin --user=michelle --clusterrole=stora
ge-admin
clusterrolebinding.rbac.authorization.k8s.io/michelle-storage-admin created

controlplane ~ ➜

controlplane ~ ➜

controlplane ~ ➜  k describe clusterrolebinding michelle-storage-admin
Name:         michelle-storage-admin
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  storage-admin
Subjects:
  Kind  Name      Namespace
  ----  ----      ---------
  User  michelle

controlplane ~ ➜  k --as michelle get storageclass
NAME                 PROVISIONER            RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSI
ON   AGE
local-path (default)   rancher.io/local-path   Delete          WaitForFirstConsumer   false
     21m

controlplane ~ ➜
```

Service Accounts

Tuesday, July 2, 2024 11:07 PM

```
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>
```

Accounts used by a application are called a service account. eg jenkins uses service account to deploy appication on kebernetes cluster

```
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>
```

Secret

```
token:
eyJhbGciOiJSUzI1NiIsImtp
ZCI6IiJ9.eyJpc3MiOiJrdWJ
lcm5ldGVzL3NlcnZpY2VhY2N
vdW50Iiwia3ViZXJuZXRlcy5
pby9zZXJ2aWNlYWNjb3Vud….
```

```
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.eyJpc3MiOiJrdWJlcm5ldGVzL
3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3V
udC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3
```

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

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-kubernetes-dashboard
spec:
  containers:
    - name: my-kubernetes-dashboard
      image: my-kubernetes-dashboard
  serviceAccountName: dashboard-sa
```

```
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 dashboard-sa-token-kbbdm (ro)
Conditions:
  Type           Status
Volumes:
  dashboard-sa-token-kbbdm:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  dashboard-sa-token-kbbdm
    Optional:    false
```

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

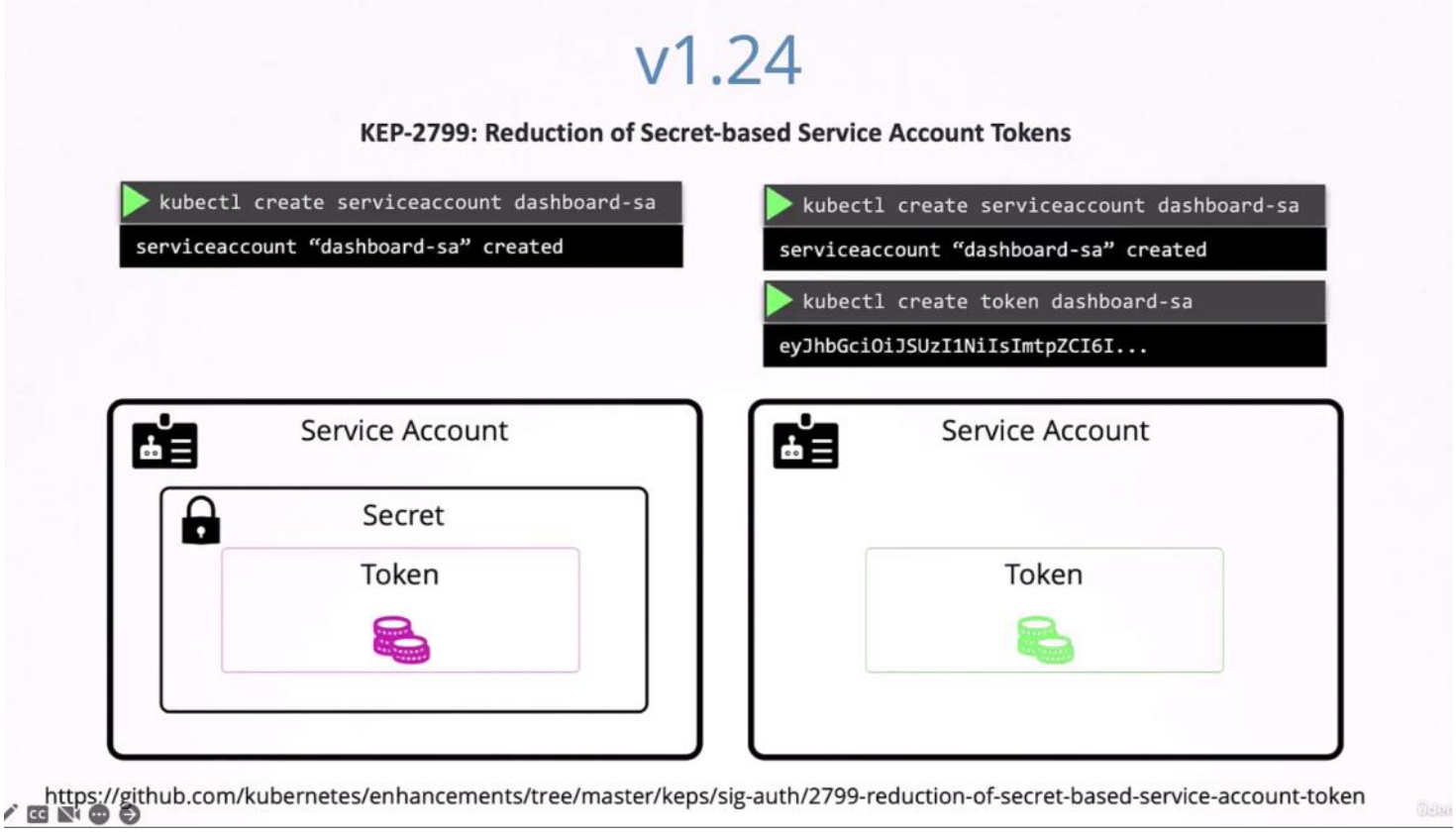
pod-definition.yml

```yaml
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
```

```
kubectl get pod my-kubernetes-dashboard -o yaml
```

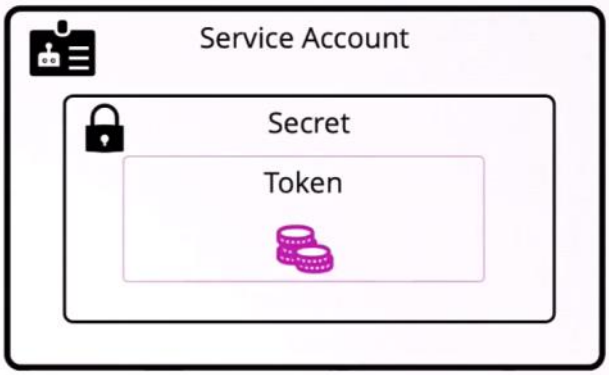
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: default
spec:
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-6mtg8
      readOnly: true
 volumes:
  - name: kube-api-access-6mtg8
    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
```



v1.24

secret-definition.yml

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



Practice- service account

Tuesday, July 2, 2024 11:18 PM

1 2 3 4 5 6 7 8 9 10 11 12 13 14

How many Service Accounts exist in the default namespace?

- 0
- 3
- 2
- 1
- 4

```
Welcome to the KodeKloud Hands-On lab

All rights reserved

controlplane ~ ➜ kubectl get sa
NAME      SECRETS   AGE
default   0         20m
dev       0         35s

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

What is the secret token used by the default service account?

- none
- default-token-zf6x9
- default-token-87x8w
- default-token-j4hkv

```
Welcome to the KodeKloud Hands-On lab

All rights reserved

controlplane ~ ➜ kubectl get sa
NAME      SECRETS   AGE
default   0         20m
dev       0         35s

controlplane ~ ➜ kubectl describe sa default
Name:                default
Namespace:           default
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   <none>
Tokens:              <none>
Events:              <none>

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

We just deployed the Dashboard application. Inspect the deployment. What is the image used by the deployment?

- kodekloud/my-kubernetes-dashboard
- gcr.io/kodekloud/customimage/my-kubernetes-dashboard
- kubernetes-dashboard
- web-dashboard

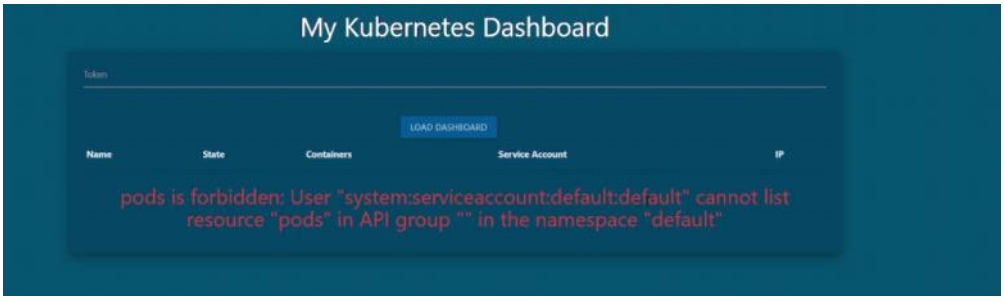
```
controlplane ~ ➜ kubectl get deployment
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
web-dashboard   1/1     1            1           20s

controlplane ~ ➜ kubectl describe deployment web-dashboard
Name:                   web-dashboard
Namespace:              default
CreationTimestamp:      Wed, 26 Jul 2023 22:41:47 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               name=web-dashboard
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  name=web-dashboard
  Containers:
   web-dashboard:
    Image:      gcr.io/kodekloud/customimage/my-kubernetes-dashboard
    Port:       8080/TCP
    Host Port:  0/TCP
    Environment:
      PYTHONUNBUFFERED:  1
    Mounts:              <none>
  Volumes:               <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   web-dashboard-97c9c59f6 (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

What type of account does the Dashboard application use to query the Kubernetes API?

- LDAP Account
- User Account
- Service Account
- OAuth Account


```
controlplane ~ ➜ kubectl get deployment
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
web-dashboard   1/1     1            1           20s

controlplane ~ ➜ kubectl describe deployment web-dashboard
Name:                   web-dashboard
Namespace:              default
CreationTimestamp:      Wed, 26 Jul 2023 22:41:47 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               name=web-dashboard
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  name=web-dashboard
  Containers:
   web-dashboard:
    Image:      gcr.io/kodekloud/customimage/my-kubernetes-dashboard
    Port:       8080/TCP
    Host Port:  0/TCP
    Environment:
      PYTHONUNBUFFERED:  1
    Mounts:              <none>
  Volumes:               <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   web-dashboard-97c9c59f6 (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

Inspect the Dashboard Application POD and identify the Service Account mounted on it.

default

service-account

dashboard-sa

```
to 1

controlplane ~ ➜

controlplane ~ ➜ kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
web-dashboard-97c9c59f6-g2zwl       1/1     Running   0          3m10s

controlplane ~ ➜ kubectl describe web-dashboard-97c9c59f6-g2zwl
error: the server doesn't have a resource type "web-dashboard-97c9c59f6-g2zwl"

controlplane ~ ✗ kubectl describe pod web-dashboard-97c9c59f6-g2zwl
Name:             web-dashboard-97c9c59f6-g2zwl
Namespace:        default
Priority:         0
Service Account:  default
Node:             controlplane/192.0.205.9
Start Time:       Wed, 26 Jul 2023 22:41:47 +0000
Labels:           name=web-dashboard
                  pod-template-hash=97c9c59f6
Annotations:      <none>
Status:           Running
IP:               10.42.0.9
IPs:
  IP:           10.42.0.9
Controlled By:  ReplicaSet/web-dashboard-97c9c59f6
Containers:
  web-dashboard:
    Container ID:   containerd://c0295f46e7c855519075930f18fff25bbebdb61e5806f01461c1b562cc77154f
    Image:          gcr.io/kodekloud/customimage/my-kubernetes-dashboard
    Image ID:       gcr.io/kodekloud/customimage/my-kubernetes-dashboard@sha256:7d70abe342b13ff1c4242dc8
3271ad73e4eedb04e2be0dd30ae7ac8852193069
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Wed, 26 Jul 2023 22:41:52 +0000
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

At what location is the ServiceAccount credentials available within the pod?

/opt/run/secrets

/var/run/secrets

/var/run/token

/etc/run/secrets

```
Containers:
  web-dashboard:
    Container ID:   containerd://c0295f46e7c855519075930f18fff25bbebdb61e5806f01461c1b562cc77154f
    Image:          gcr.io/kodekloud/customimage/my-kubernetes-dashboard
    Image ID:       gcr.io/kodekloud/customimage/my-kubernetes-dashboard@sha256:7d70abe342b13ff1c4242dc8
3271ad73e4eedb04e2be0dd30ae7ac8852193069
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Wed, 26 Jul 2023 22:41:52 +0000
    Ready:          True
    Restart Count:  0
    Environment:
      PYTHONUNBUFFERED:  1
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-swjvh (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-swjvh:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

The application needs a ServiceAccount with the Right permissions to be created to authenticate to Kubernetes. The `default` ServiceAccount has limited access. Create a new ServiceAccount named `dashboard-sa`.

Service Account Name: dashboard-sa

```
controlplane ~ ➜ kubectl create serviceaccount dashboard-sa
serviceaccount/dashboard-sa created

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

We just added additional permissions for the newly created `dashboard-sa` account using RBAC.

If you are interested checkout the files used to configure RBAC at `/var/rbac`. We will discuss RBAC in a separate section.

Ok

```
controlplane ~ ➜ kubectl create serviceaccount dashboard-sa
serviceaccount/dashboard-sa created

controlplane ~ ➜ cd /var/rbac/

controlplane /var/rbac ➜ ls
dashboard-sa-role-binding.yaml  pod-reader-role.yaml

controlplane /var/rbac ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

Enter the access token in the UI of the dashboard application. Click `Load Dashboard` button to load Dashboard

Create an authorization token for the newly created service account, copy the generated token and paste it into the `token` field of the UI.

To do this, run `kubectl create token dashboard-sa` for the `dashboard-sa` service account, copy the token and paste it in the UI.

Ok

```
controlplane ~ ➜ kubectl create serviceaccount dashboard-sa
serviceaccount/dashboard-sa created

controlplane ~ ➜ cd /var/rbac/

controlplane /var/rbac ➜ ls
dashboard-sa-role-binding.yaml  pod-reader-role.yaml

controlplane /var/rbac ➜ kubectl create token dashboard-sa
eyJhbGciOiJSUzI1NiIsImtpZCI6InU5U0pKRmthaThHcTBjZ0lLRkdaZHl5RjBFUE1wMkxmR3RDODlnck9iRTQifQ.eyJhdWQiOlsia
HR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiLCJrM3MiXSwiZXhwIjoxNjkwNDE1MjkzLCJpYXQiOjE2O
TA0MTE2OTMsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZXRlcy5pbyI6e
yJuYW1lc3BhY2UiOiJkZWZhdWx0Iiwic2Vydmljj ZWFjY291bnQiOnsibmFtZSI6ImRhc2hib2FyZC1zYSIsInVpZCI6IjNiYmVjZjcx
Tc2OTYtNGFmYi1hZmY5LTEzM2M1NDJlZGM4NCJ9fSwibmJmIjoxNjkwNDExNjkzLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ
GVmYXVsdDpkYXNoYm9hcmQtc2EifQ.VGp7Jz-7ycG1uyvE7CBbIQauMNRmoxLg9bentGkEVzw2ILBoZI1C3SmWPkZmybZdXPTcBpiK
qas0cQ1QrLI477kazUPLwHaoMHRmHpaDYzmVNdLaqUmgm34DMXPrjd2iNd5e2RBLqeHvJL6dulD-FtPTChLkIZsFYKetnqvZrCsnfJ
HQKNgrW_Ui6TeIs2-CAjkoFrC2VDHMItScQsx-ocKGmdMR1_Z
32GA72Biqi2cBAx3IpRLdUGWYbxybHhDrqkA10A9wDaIoj8Hd
Czj4g

controlplane /var/rbac ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13

14

You shouldn't have to copy and paste the token each time. The Dashboard application is programmed to read token from the secret mount location. However currently, the `default` service account is mounted. Update the deployment to use the newly created ServiceAccount

Edit the deployment to change ServiceAccount from `default` to `dashboard-sa`.

Check

- Deployment name: web-dashboard
- Service Account: dashboard-sa
- Deployment Ready

```
controlplane ~ ➜ kubectl create serviceaccount dashboard-sa
serviceaccount/dashboard-sa created

controlplane ~ ➜ cd /var/rbac/

controlplane /var/rbac ➜ ls
dashboard-sa-role-binding.yaml  pod-reader-role.yaml

controlplane /var/rbac ➜ kubectl create token dashboard-sa
eyJhbGciOiJSUzI1NiIsImtpZCI6InU5U0pKRmthaThHcTBjZ0lLRkdaZHl5Rj8FUE1wMkxmR3RDODlnck9iRTQifQ.eyJhdWQiOls
HR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiLCJrM3MiXSwiZXhwIjoxNjkwNDE1MjkzLCJpYXQiOjE
TA0MTE2OTMsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZXRlcy5pbyI
yJuYW1lc3BhY2UiOiJkZWZhdWx0Iiwic2VydmljZWFjY291bnQiOnsibmFtZSI6ImRhc2hib2FyZC1zYSIsInVpZCI6IjNiYmVjZjc
Tc2OTYtNGFmYi1hZmY5LTEzM2M1NDJlZGM4NCJ9fSwibmJmIjoxNjkwNDExNjkzLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ
GVmYXVsdDpkYXNoYm9hcmQtc2EifQ.VGp7Jz-7ycG1uyvE7CBbIQauMNRmoxLg9bentGkEVzw2ILBoZI1C3SmWPkZmybZdXPTcBpik
qas0cQ1QrLI477kazUPLwHaoMHRmHpaDYzmVNdLaqUmgm34DMXPrjd2iNd5e2RBLqeHvJL6dulD-FtPTChLkIZsFYKetnqvZrCsnfJ
HQKNgrW_Ui6TeIs2-CAjkoFrC2VDHMItScQsx-ocKGmdMR1_ZdqfgR5G5O11B3RJRlIrXz2xpWC4yiW78jUl9cLTKVa86MoNyKkrNg
32GA72Biqi2cBAx3IpRLdUGWYbxybHhDrqkA10A9wDaIoj8Hd140LXkCzj4g

controlplane /var/rbac ➜ kubectl get deployment
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
web-dashboard   1/1     1            1           7m32s

controlplane /var/rbac ➜ kubectl get deployment web-dashboard -o yaml > dashboard.yaml

controlplane /var/rbac ➜ vi dashboard.yaml

controlplane /var/rbac ➜ kubec
```

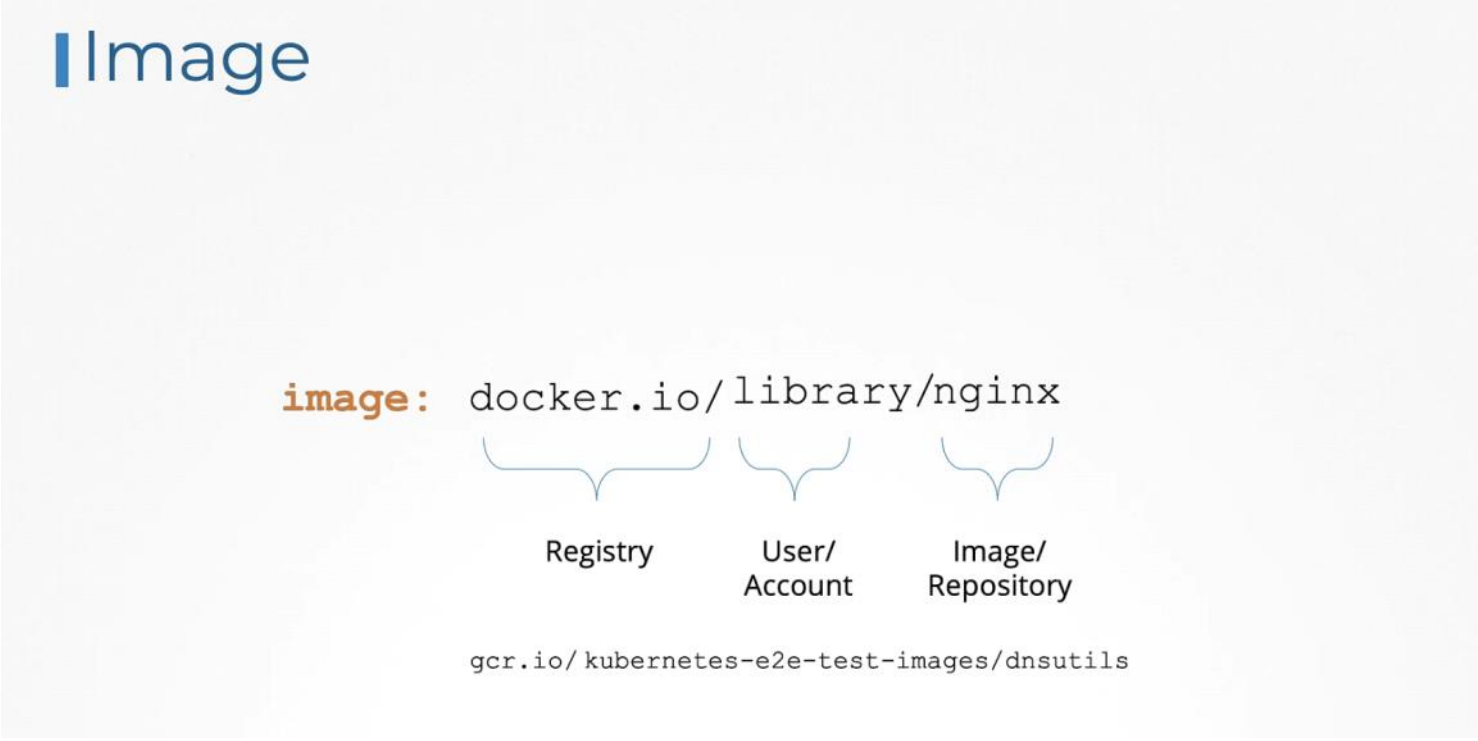
```yaml
  generation: 1
  name: web-dashboard
  namespace: default
  resourceVersion: "1024"
  uid: daaa0628-d1bd-4624-b5e3-9475f1958464
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      name: web-dashboard
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        name: web-dashboard
    spec:
      serviceAccountName: dashboard-sa
      containers:
      - env:
        - name: PYTHONUNBUFFERED
          value: "1"
        image: gcr.io/kodekloud/customimage/my-kubernetes-dashboard
        imagePullPolicy: Always
        name: web-dashboard
        ports:
        - containerPort: 8080
          protocol: TCP
        resources: {}
-- INSERT (paste) --
```

```
controlplane /var/rbac ➜ vi dashboard.yaml

controlplane /var/rbac ➜ kubectl apply -f dashboard.yaml
Warning: resource deployments/web-dashboard is missing the kubectl.kubernetes.io/last-applied-configurat
ion annotation which is required by kubectl apply. kubectl apply should only be used on resources create
d declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be
patched automatically.
deployment.apps/web-dashboard configured

controlplane /var/rbac ➜
```

```
controlplane /var/rbac ➜ kubectl apply -f dashboard.yaml
Warning: resource deployments/web-dashboard is missing the kubectl.kubernetes.io/last-applied-configurat
ion annotation which is required by kubectl apply. kubectl apply should only be used on resources create
d declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be
patched automatically.
deployment.apps/web-dashboard configured

controlplane /var/rbac ➜ kubectl get deployment
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
web-dashboard   1/1     1            1           8m50s

controlplane /var/rbac ➜
```

Image security

Wednesday, July 3, 2024 12:32 AM



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
```

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

Private Repository

```
docker login private-registry.io
```

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

nginx-pod.yaml

```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
```

Practice- image security

Wednesday, July 3, 2024 12:38 AM

01 02 03 04 05 06 07

What secret type must we choose for `docker registry`?

registry-docker

tls

docker-registry

generic

registry

```
root@controlplane:/#
root@controlplane:/#
root@controlplane:/# kubectl create secret
Create a secret using specified subcommand.

Available Commands:
  docker-registry Create a secret for use with a Docker registry
  generic         Create a secret from a local file, directory or literal value
  tls             Create a TLS secret

Usage:
  kubectl create secret [flags] [options]

Use "kubectl <command> --help" for more information about a given command.
Use "kubectl options" for a list of global command-line options (applies to all commands).
root@controlplane:/#
```

01 02 03 04 05 06 07

We have an application running on our cluster. Let us explore it first. What image is the application using?

nginx:alpine

webapp

nginx

busybox

```
root@controlplane:/#
root@controlplane:/#
root@controlplane:/# k describe deploy web
Name:                   web
Namespace:              default
CreationTimestamp:      Mon, 18 Apr 2022 01:22:13 +0000
Labels:                 app=web
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=web
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=web
  Containers:
   nginx:
    Image:        nginx:alpine
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   web-bd975bd87 (2/2 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  47s   deployment-controller  Scaled up replica set web-bd975bd87 to 2
root@controlplane:/#
```

01 02 03 04 05 06 07

We decided to use a modified version of the application from an internal private registry. Update the image of the deployment to use a new image from

`myprivateregistry.com:5000`

The registry is located at `myprivateregistry.com:5000`. Don't worry about the credentials for now. We will configure them in the upcoming steps.

Check

● Use Image from private registry

```
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: "2022-04-18T01:22:13Z"
  generation: 1
  labels:
    app: web
  name: web
  namespace: default
  resourceVersion: "3173"
  uid: fdc6a23a-fc56-46ed-89c5-b102c6c0fe36
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: web
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: web
    spec:
      containers:
      - image: myprivateregistry.com:5000/nginx:alpine
        imagePullPolicy: IfNotPresent
        name: nginx
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
                                                          37,42
```

```
root@controlplane:/# k edit deploy web
deployment.apps/web edited
root@controlplane:/#
root@controlplane:/#
root@controlplane:/# k describe deploy web
```

01 02 03 04 05 06 07

Are the new PODs created with the new images successfully running?

Yes

No

```
root@controlplane:/#
```

01 02 03 04 05 06 07

Create a secret object with the credentials required to access the registry.

Name: `private-reg-cred`

Username: `dock_user`

Password: `dock_password`

Server: `myprivateregistry.com:5000`

Email: `dock_user@myprivateregistry.com`

```
root@controlplane:/# #kubectl create secret docker-registry my-secret --docker-server=DOCKER_REGISTRY_SE
RVER --docker-username=DOCKER_USER --docker-password=DOCKER_PASSWORD --docker-email=DOCKER_EMAIL
root@controlplane:/#
root@controlplane:/#
root@controlplane:/#
root@controlplane:/# #kubectl create secret docker-registry private-reg-cred --docker-server=myprivatere
gistry.com:5000 --docker-username=dock_user --docker-password=dock_password --docker-email=dock_user@myp
rivateregistry.com
root@controlplane:/# kubectl create secret docker-registry private-reg-cred --docker-server=myprivatereg
istry.com:5000 --docker-username=dock_user --docker-password=dock_password --docker-email=dock_user@mypr
ivateregistry.com
secret/private-reg-cred created
root@controlplane:/#
root@controlplane:/#
root@controlplane:/#
```

01 02 03 04 05 06 07

Configure the deployment to use credentials from the new secret to pull images from the private registry

Check

Image Pull Secret: private-reg-cred

```
root@controlplane:/# #kubectl create secret docker-registry my-secret --docker-server=DOCKER_REGISTRY
RVER --docker-username=DOCKER_USER --docker-password=DOCKER_PASSWORD --docker-email=DOCKER_EMAIL
root@controlplane:/#
root@controlplane:/#
root@controlplane:/#
root@controlplane:/# #kubectl create secret docker-registry private-reg-cred --docker-server=myprivat
gistry.com:5000 --docker-username=dock_user --docker-password=dock_password --docker-email=dock_user
rivateregistry.com
root@controlplane:/# kubectl create secret docker-registry private-reg-cred --docker-server=myprivat
istry.com:5000 --docker-username=dock_user --docker-password=dock_password --docker-email=dock_user@
ivateregistry.com
secret/private-reg-cred created
root@controlplane:/#
root@controlplane:/#
root@controlplane:/#
root@controlplane:/# k edit deploy web
deployment.apps/web edited
root@controlplane:/#
```

```
  uid: fdc6a23a-fc56-46ed-89c5-b102c6c0fe36
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: web
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: web
    spec:
      imagePullSecrets:
      - name: private-reg-cred
      containers:
      - image: myprivateregistry.com:5000/nginx:alpine
        imagePullPolicy: IfNotPresent
        name: nginx
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
status:
  availableReplicas: 2
  conditions:
  - lastTransitionTime: "2022-04-18T01:22:35Z"
    lastUpdateTime: "2022-04-18T01:22:35Z"
-- INSERT --
```

01 02 03 04 05 06 07

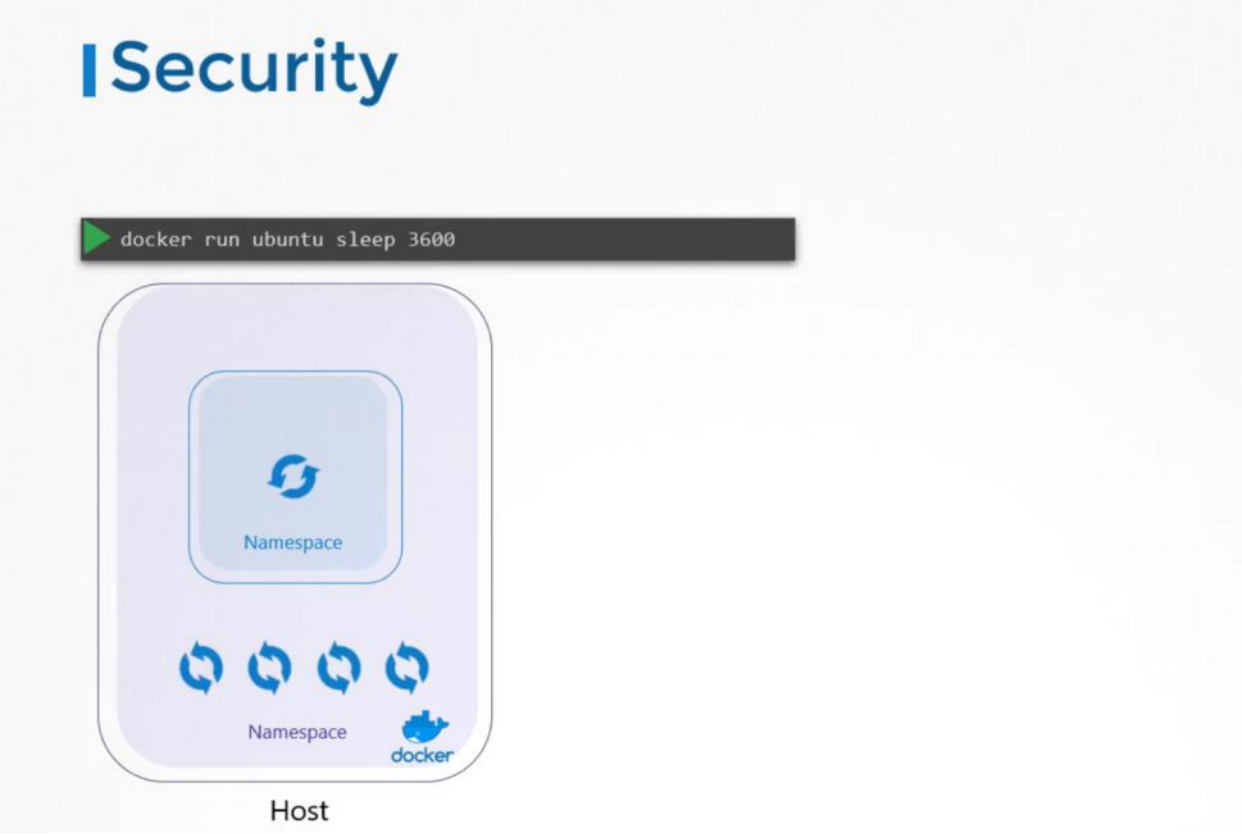
Check the status of PODs. Wait for them to be running. You have now successfully configured a Deployment to pull images from the private registry.

Ok

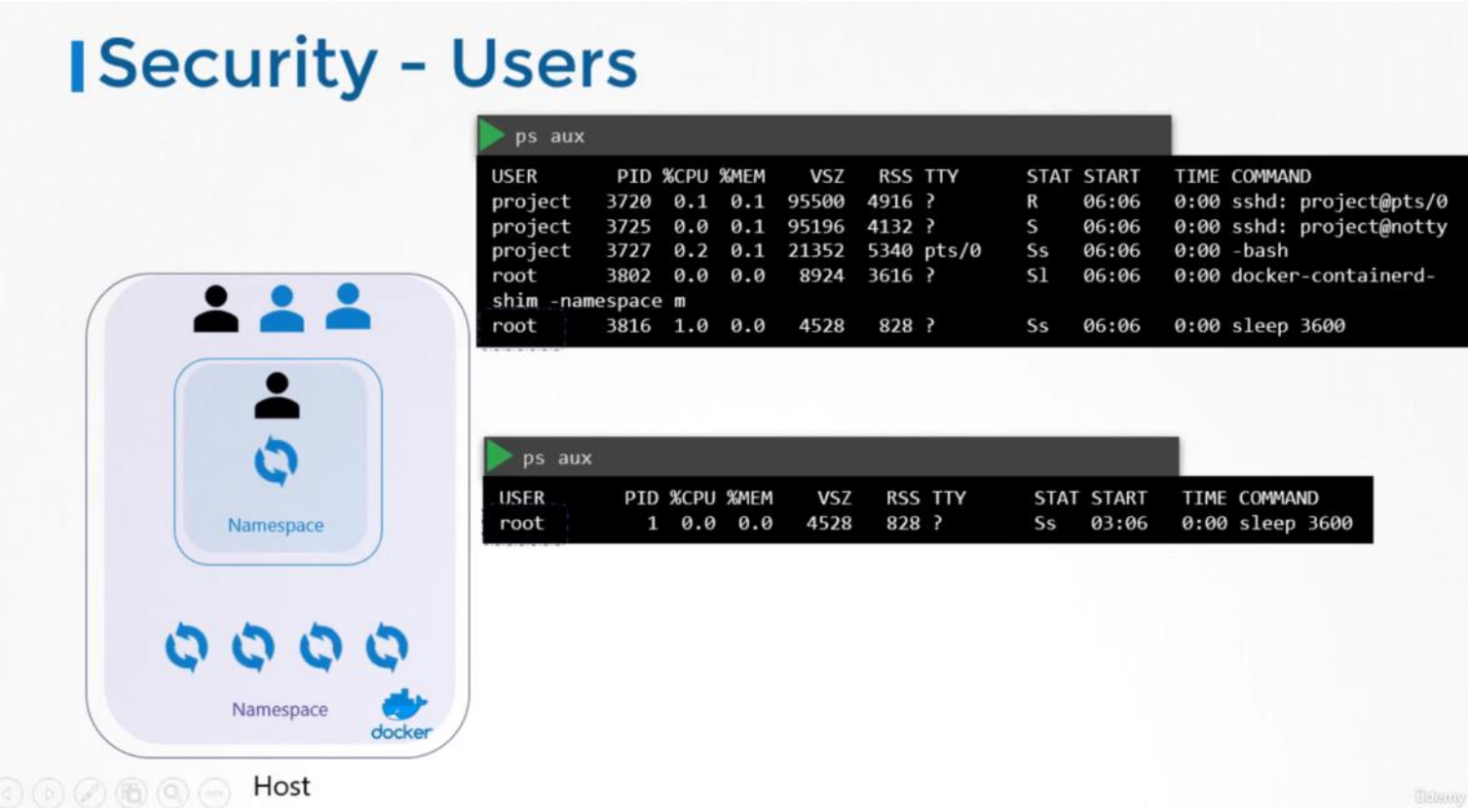
```
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   web-5f558bb687 (2/2 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  6m22s  deployment-controller  Scaled up replica set web-bd975bd87 to 2
  Normal  ScalingReplicaSet  4m42s  deployment-controller  Scaled up replica set web-85fcf65896 to 1
  Normal  ScalingReplicaSet  8s     deployment-controller  Scaled down replica set web-85fcf65896 to 0
  Normal  ScalingReplicaSet  8s     deployment-controller  Scaled up replica set web-5f558bb687 to 1
  Normal  ScalingReplicaSet  6s     deployment-controller  Scaled down replica set web-bd975bd87 to 1
  Normal  ScalingReplicaSet  6s     deployment-controller  Scaled up replica set web-5f558bb687 to 2
  Normal  ScalingReplicaSet  4s     deployment-controller  Scaled down replica set web-bd975bd87 to 0
root@controlplane:/#
root@controlplane:/#
root@controlplane:/#
root@controlplane:/#
root@controlplane:/# k get pods
NAME                   READY   STATUS        RESTARTS   AGE
web-5f558bb687-dxthl   1/1     Running       0          26s
web-5f558bb687-qprgq   1/1     Running       0          28s
web-85fcf65896-rbsmq   0/1     Terminating   0          5m2s
root@controlplane:/#
root@controlplane:/#
root@controlplane:/# k get pods
NAME                   READY   STATUS        RESTARTS   AGE
web-5f558bb687-dxthl   1/1     Running       0          28s
web-5f558bb687-qprgq   1/1     Running       0          30s
web-85fcf65896-rbsmq   0/1     Terminating   0          5m4s
root@controlplane:/# k get pods
NAME                   READY   STATUS        RESTARTS   AGE
web-5f558bb687-dxthl   1/1     Running       0          32s
web-5f558bb687-qprgq   1/1     Running       0          34s
web-85fcf65896-rbsmq   0/1     Terminating   0          5m8s
root@controlplane:/#
```

Security in docker

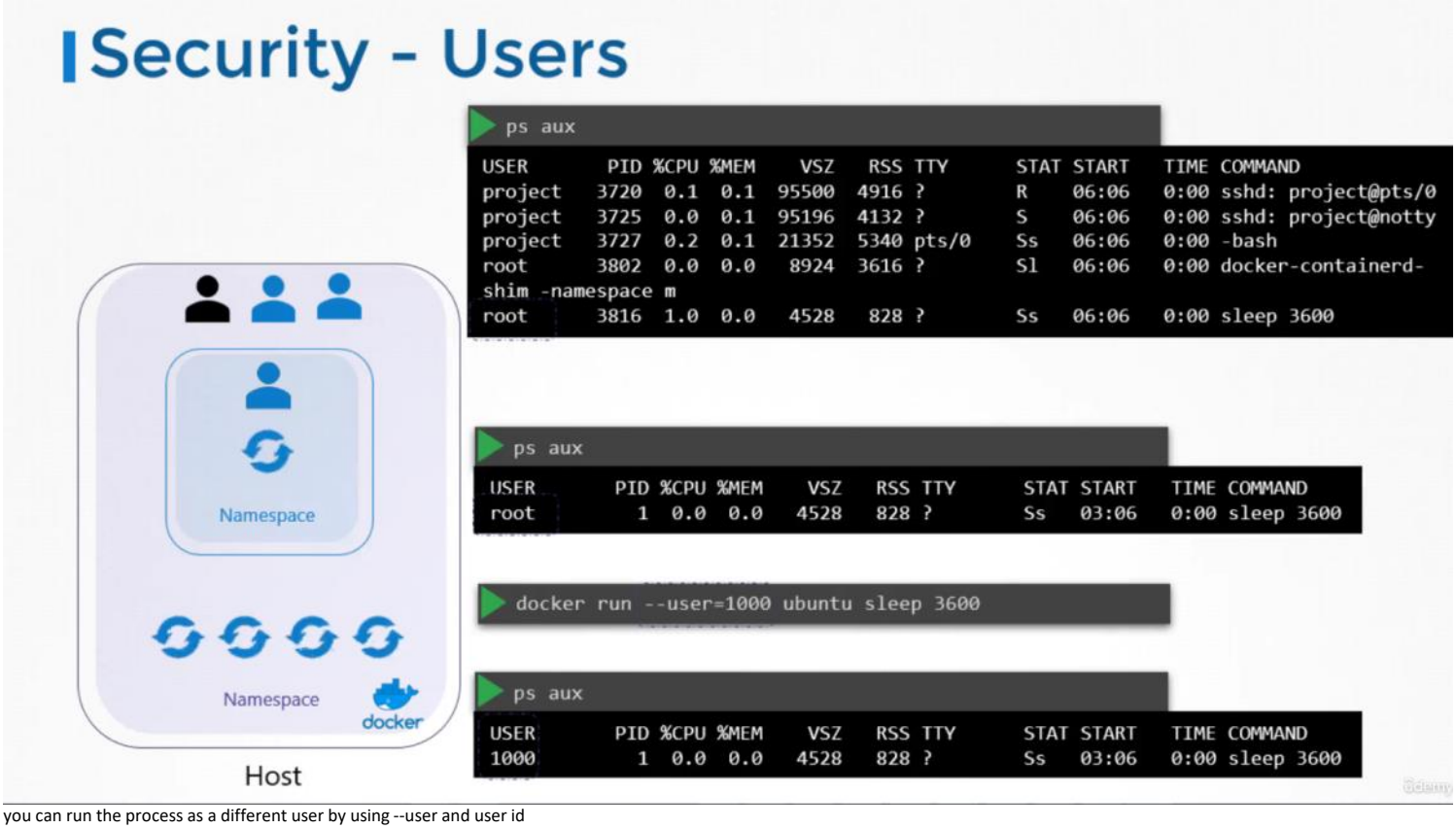
Wednesday, July 3, 2024 10:39 AM



a docker container can only see processes from within its namespace.
processes can have different process ID in different namespaces

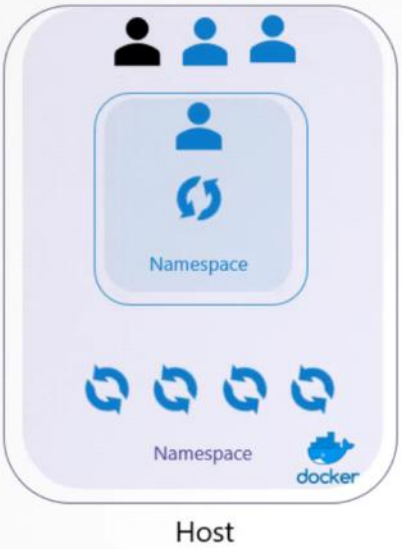


by default docker run processes ina container as root user.



you can run the process as a different user by using --user and user id

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 ?        Ss   03:06   0:00 sleep 3600
```

Linux Capabilities

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

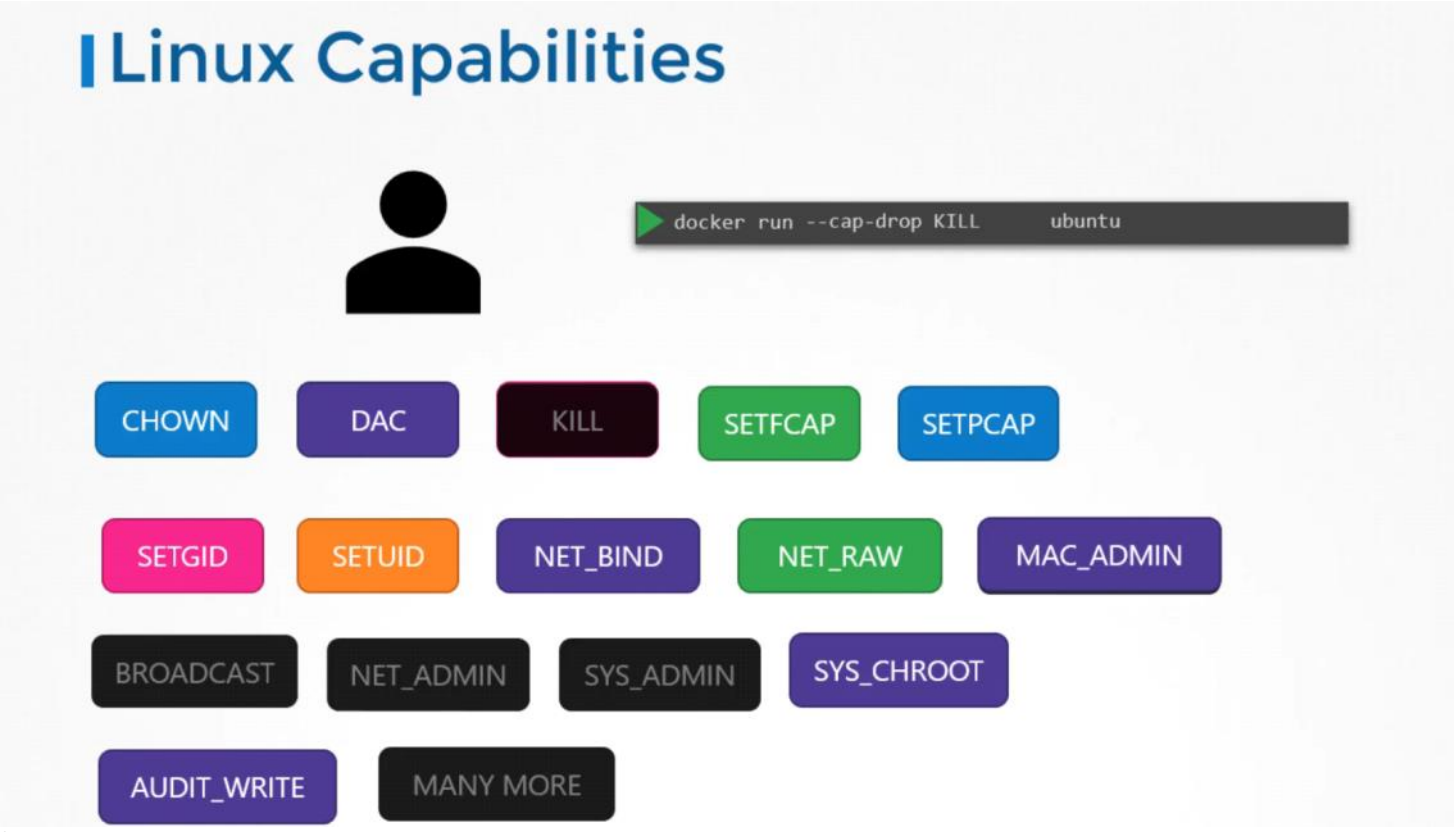
CHOWN DAC KILL SETFCAP SETPCAP

SETGID SETUID NET_BIND NET_RAW MAC_ADMIN

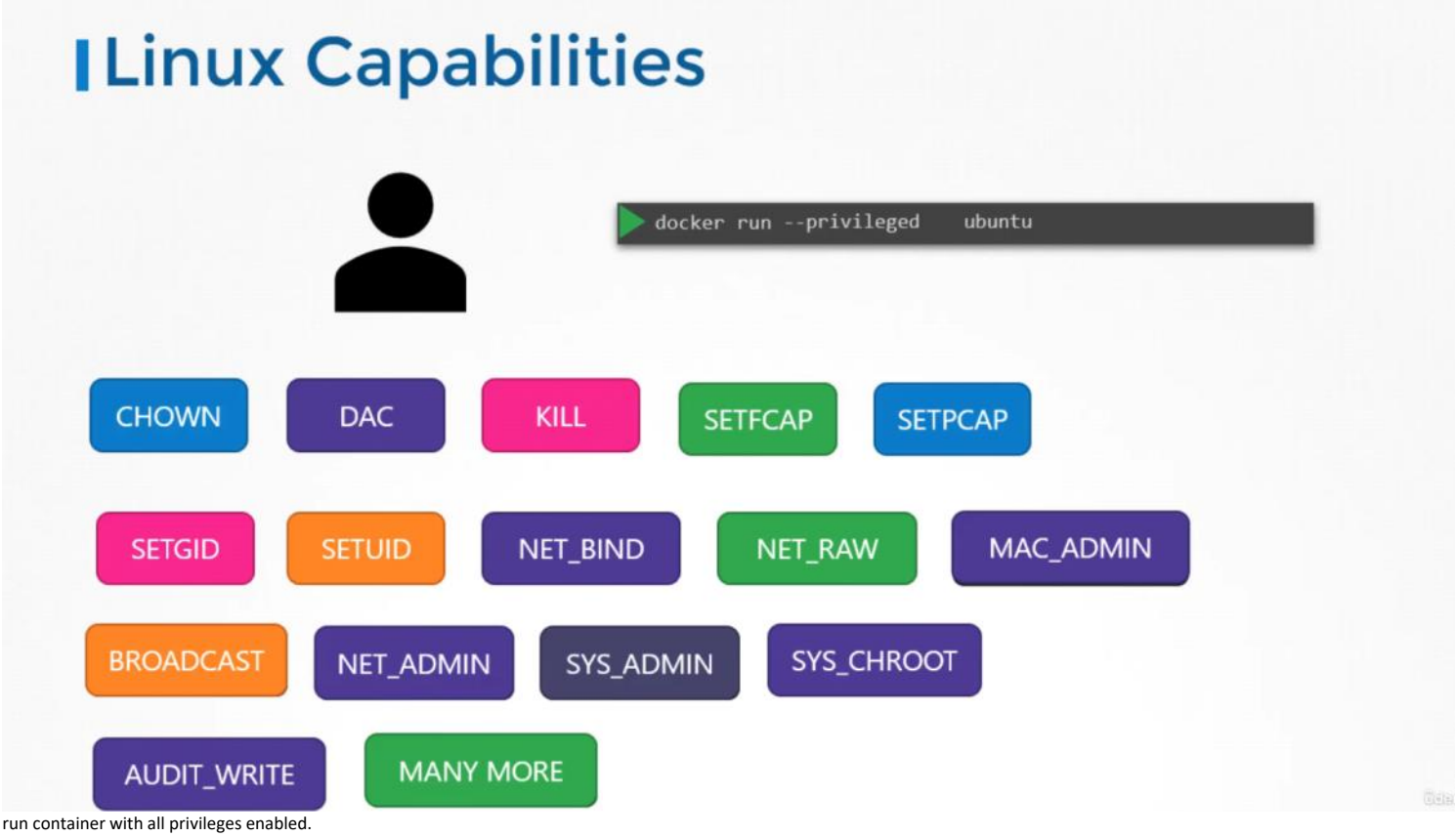
BROADCAST NET_ADMIN SYS_ADMIN SYS_CHROOT

AUDIT_WRITE MANY MORE

user the cap add command to give privileges to container to perform actions that can reboot host, make changes that can disrupt the host or other containers running on same host



for dropping privileges



run container with all privileges enabled.

Security context

Wednesday, July 3, 2024 10:52 AM

Security Context

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

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

configuration set on pod level

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

configuration set on pod level

Practice- security context

Wednesday, July 3, 2024 10:59 AM

1 2 3 4 5 6

What is the user used to execute the sleep process within the `ubuntu-sleeper` pod?

In the current(default) namespace.

- user2
- test-user
- user1
- root
- my-user

```
controlplane ~ ➜ whoami
root

controlplane ~ ➜ kubectl get pod
NAME             READY   STATUS    RESTARTS   AGE
ubuntu-sleeper   1/1     Running   0          7m58s

controlplane ~ ➜ kubectl exec ubuntu-sleepr -- whoami
Error from server (NotFound): pods "ubuntu-sleepr" not found

controlplane ~ ✗ kubectl exec ubuntu-sleeper -- whoami
root

controlplane ~ ➜
```

1 2 3 4 5 6

Edit the pod `ubuntu-sleeper` to run the sleep process with user ID `1010`.

Note: Only make the necessary changes. Do not modify the name or image of the pod.

Check

- Pod Name: ubuntu-sleeper
- Image Name: ubuntu
- SecurityContext: User 1010

```
Welcome to the KodeKloud Hands-On lab

All rights reserved

controlplane ~ ➜ kubectl get pod ubunut-sleeper -o yaml > ubuntu-sleeper.yaml
Error from server (NotFound): pods "ubunut-sleeper" not found

controlplane ~ ✗ kubectl get pod ubuntu-sleeper -o yaml > ubuntu-sleeper.yaml

controlplane ~ ➜ vi ubuntu-sleeper.yaml

controlplane ~ ➜ kubectl delete pod ubuntu-sleeper --force
Warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource may co
ntinue to run on the cluster indefinitely.
pod "ubuntu-sleeper" force deleted

controlplane ~ ➜ k
```

1 2 3 4 5 6

Edit the pod `ubuntu-sleeper` to run the sleep process with user ID `1010`.

Note: Only make the necessary changes. Do not modify the name or image of the pod.

Check

- Pod Name: ubuntu-sleeper
- Image Name: ubuntu
- SecurityContext: User 1010

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2023-07-26T22:06:23Z"
  name: ubuntu-sleeper
  namespace: default
  resourceVersion: "935"
  uid: 07c53385-6987-40a0-a5c3-93282649b4a0
spec:
  containers:
  - command:
    - sleep
    - "4800"
    image: ubuntu
    imagePullPolicy: Always
    name: ubuntu
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-wvn5z
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: controlplane
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext:
    runAsUser: 1010
```

```
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext:
    runAsUser: 1010
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
-- INSERT (paste) --
```

1 2 3 4 5 6

Edit the pod `ubuntu-sleeper` to run the sleep process with user ID `1010`.

Note: Only make the necessary changes. Do not modify the name or image of the pod.

...

- Pod Name: ubuntu-sleeper
- Image Name: ubuntu
- SecurityContext: User 1010

```
      Welcome to the KodeKloud Hands-On lab

            All rights reserved

controlplane ~ ➜ kubectl get pod ubunut-sleeper -o yaml > ubuntu-sleeper.yaml
Error from server (NotFound): pods "ubunut-sleeper" not found

controlplane ~ ✗ kubectl get pod ubuntu-sleeper -o yaml > ubuntu-sleeper.yaml

controlplane ~ ➜ vi ubuntu-sleeper.yaml

controlplane ~ ➜ kubectl delete pod ubuntu-sleeper --force
Warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource may co
ntinue to run on the cluster indefinitely.
pod "ubuntu-sleeper" force deleted

controlplane ~ ➜ kubectl apply -f ubuntu-sleeper.yaml
pod/ubuntu-sleeper created

controlplane ~ ➜ kubectl get pod
NAME             READY   STATUS    RESTARTS   AGE
ubuntu-sleeper   1/1     Running   0          4s

controlplane ~ ➜
```

```
controlplane ~ ➜ vi multi-pod.yaml
```

1 2 3 4 5 6

A Pod definition file named `multi-pod.yaml` is given. With what user are the processes in the `web` container started?

The pod is created with multiple containers and security contexts defined at the `Pod` and `Container` level.

- 1002
- 1000
- 1001
- root

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-pod
spec:
  securityContext:
    runAsUser: 1001
  containers:
  -  image: ubuntu
     name: web
     command: ["sleep", "5000"]
     securityContext:
      runAsUser: 1002

  -  image: ubuntu
     name: sidecar
     command: ["sleep", "5000"]
```

here container level specifications will override the pod level specifications.

1 2 3 4 5 6

With what user are the processes in the `sidecar` container started?

The pod is created with multiple containers and security contexts defined at the `Pod` and `Container` level.

- root

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-pod
spec:
  securityContext:
    runAsUser: 1001
  containers:
  -  image: ubuntu
     name: web
     command: ["sleep", "5000"]
     securityContext:
      runAsUser: 1002

  -  image: ubuntu
```

1 2 3 4 5 6

With what user are the processes in the `sidecar` container started?

The pod is created with multiple containers and security contexts defined at the `Pod` and `Container` level.

- root
- 1001
- 1000
- 1002

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: multi-pod
spec:
  securityContext:
    runAsUser: 1001
  containers:
  -  image: ubuntu
     name: web
     command: ["sleep", "5000"]
     securityContext:
      runAsUser: 1002

  -  image: ubuntu
     name: sidecar
     command: ["sleep", "5000"]
```

1 2 3 4 5 6

Update pod `ubuntu-sleeper` to run as Root user and with the `SYS_TIME` capability.

Note: Only make the necessary changes. Do not modify the name of the pod.

Check

- Pod Name: ubuntu-sleeper
- Image Name: ubuntu
- SecurityContext: Capability SYS_TIME
- Is run as a root user?

```
controlplane ~ ➜ vi ubuntu-sleeper.yaml
```

1 2 3 4 5 6

Update pod `ubuntu-sleeper` to run as Root user and with the `SYS_TIME` capability.

Note: Only make the necessary changes. Do not modify the name of the pod.

Check

- Pod Name: ubuntu-sleeper
- Image Name: ubuntu
- SecurityContext: Capability SYS_TIME
- Is run as a root user?

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2023-07-26T22:12:07Z"
  name: ubuntu-sleeper
  namespace: default
  resourceVersion: "945"
  uid: 34ce000d-d278-4988-b6be-f9e41086be9f
spec:
  containers:
  - command:
    - sleep
    - "4800"
    image: ubuntu
    imagePullPolicy: Always
    name: ubuntu
    securityContext:
      capabilities:
        add: ["SYS_TIME"]
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-km279
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: controlplane
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
```

Task Hint Solution 54:54

1 2 3 4 5 6

Update pod `ubuntu-sleeper` to run as Root user and with the `SYS_TIME` capability.

Note: Only make the necessary changes. Do not modify the name of the pod.

Complete Next

- Pod Name: ubuntu-sleeper
- Image Name: ubuntu
- SecurityContext: Capability SYS_TIME
- Is run as a root user?

Terminal 1

```
controlplane ~ ➜ vi ubuntu-sleeper.yaml

controlplane ~ ➜ kubectl delete pod ubuntu-sleeper --force
Warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource
y continue to run on the cluster indefinitely.
pod "ubuntu-sleeper" force deleted

controlplane ~ ➜ kubectl apply -f ubuntu-sleeper.yaml
pod/ubuntu-sleeper created

controlplane ~ ➜
```

1 2 3 4 5 6

Now update the pod to also make use of the `NET_ADMIN` capability.

Note: Only make the necessary changes. Do not modify the name of the pod.

Check

- Pod Name: ubuntu-sleeper
- Image Name: ubuntu
- SecurityContext: Capability SYS_TIME
- SecurityContext: Capability NET_ADMIN

```
controlplane ~ ➜ vi ubuntu-sleeper.yaml

controlplane ~ ➜ kubectl delete pod ubuntu-sleeper --force
Warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource m
y continue to run on the cluster indefinitely.
pod "ubuntu-sleeper" force deleted

controlplane ~ ➜ kubectl apply -f ubuntu-sleeper.yaml
pod/ubuntu-sleeper created

controlplane ~ ➜ vi ubuntu-sleeper.yaml
```

Task Hint Solution 54:13

1 2 3 4 5 6

Now update the pod to also make use of the `NET_ADMIN` capability.

Note: Only make the necessary changes. Do not modify the name of the pod.

Check

- Pod Name: ubuntu-sleeper
- Image Name: ubuntu
- SecurityContext: Capability SYS_TIME
- SecurityContext: Capability NET_ADMIN

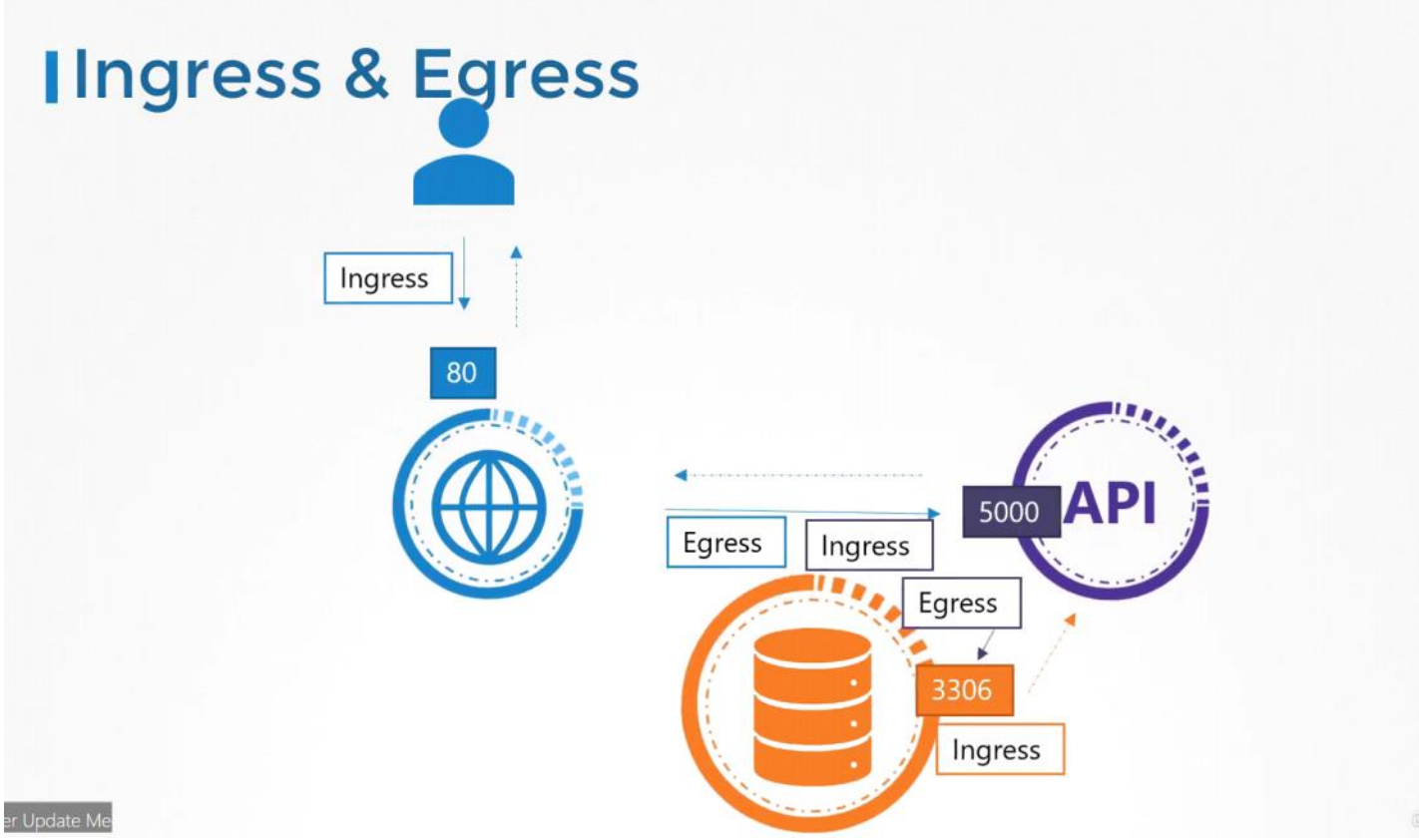
Driver Update Me

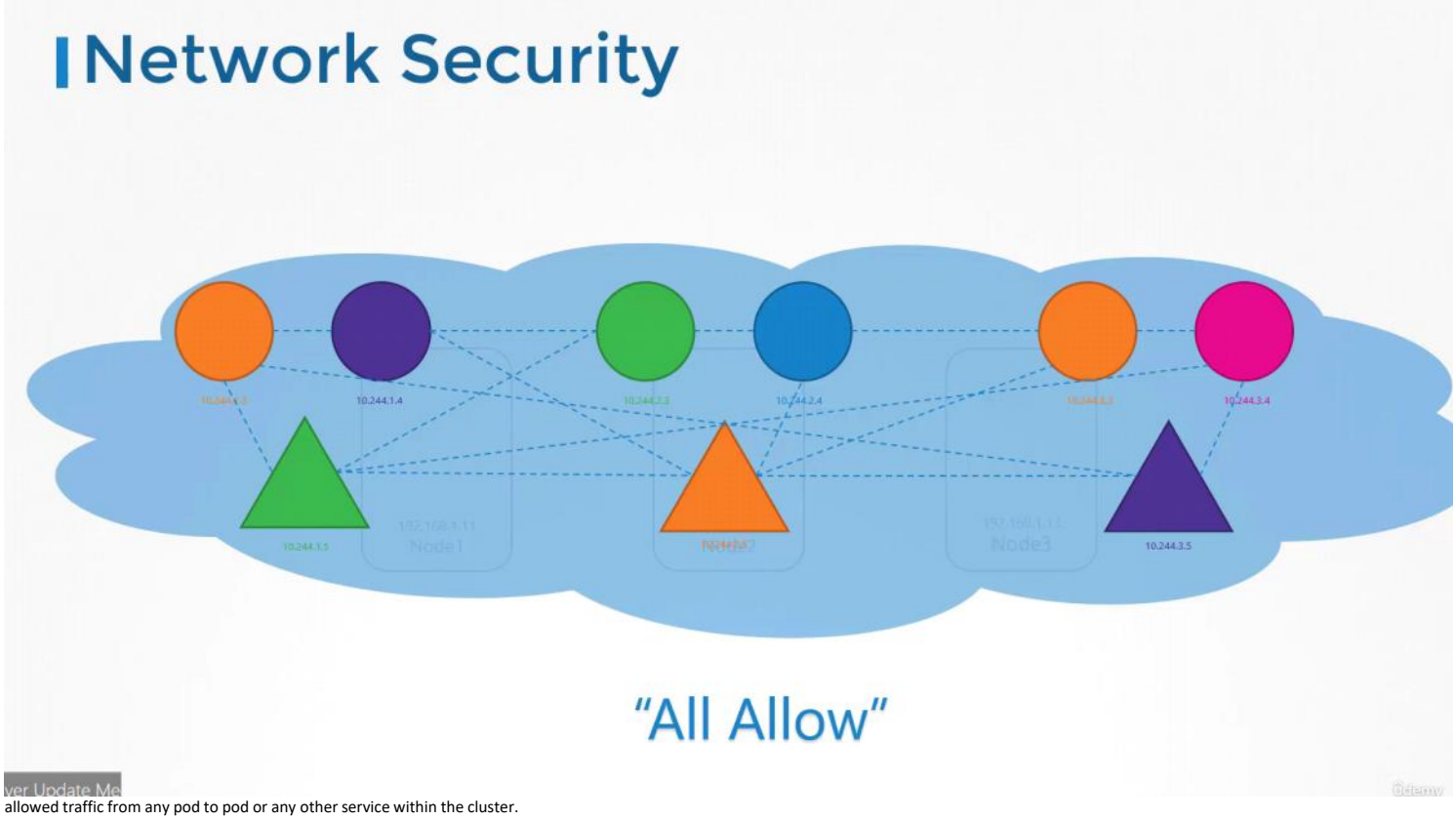
Terminal 1

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2023-07-26T22:12:07Z"
  name: ubuntu-sleeper
  namespace: default
  resourceVersion: "945"
  uid: 34ce000d-d278-4988-b6be-f9e41086be9f
spec:
  containers:
  - command:
    - sleep
    - "4800"
    image: ubuntu
    imagePullPolicy: Always
    name: ubuntu
    securityContext:
      capabilities:
        add: ["SYS_TIME", "NET_ADMIN"]
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-km279
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: controlplane
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
-- INSERT (paste) --
```

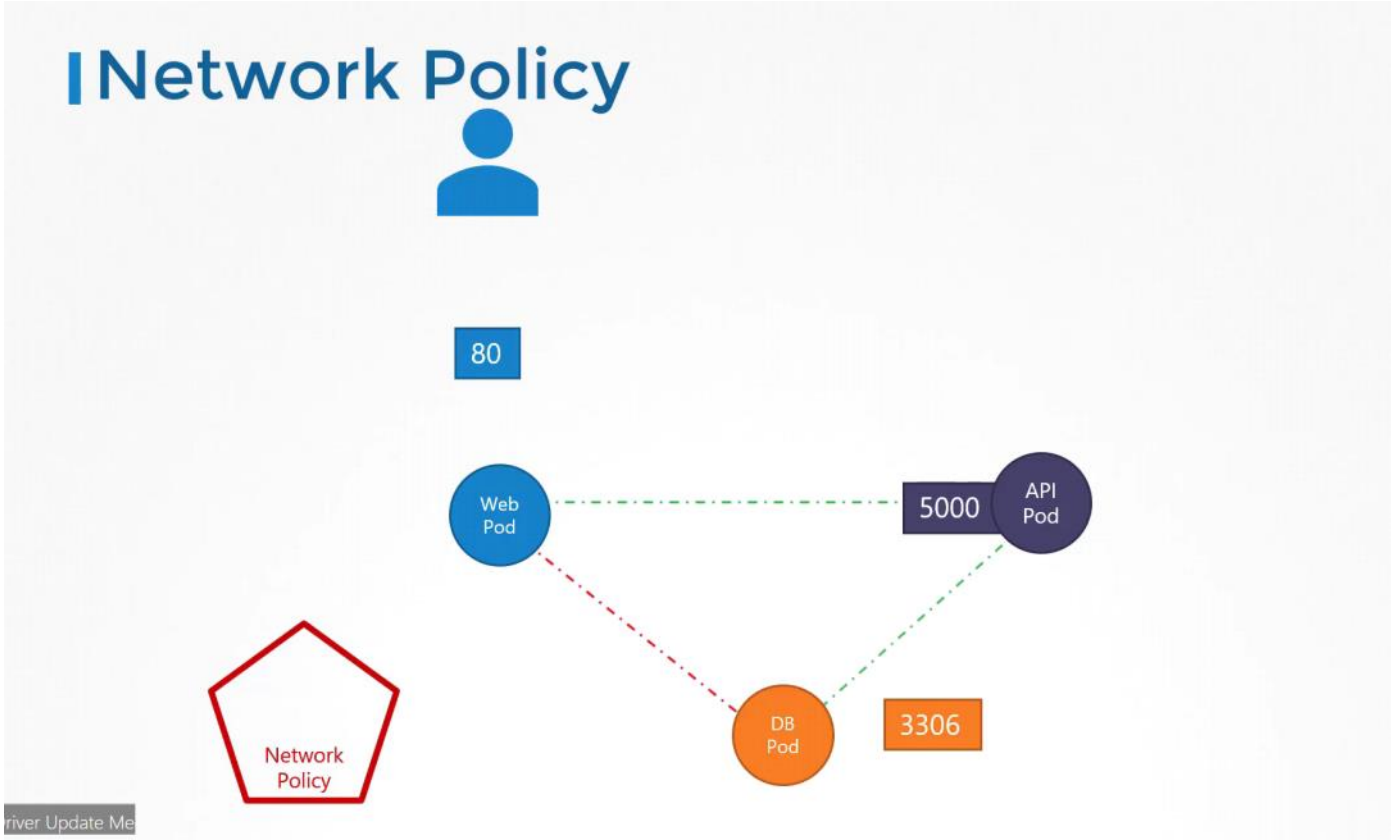
Network Policies

Friday, July 5, 2024 8:30 AM

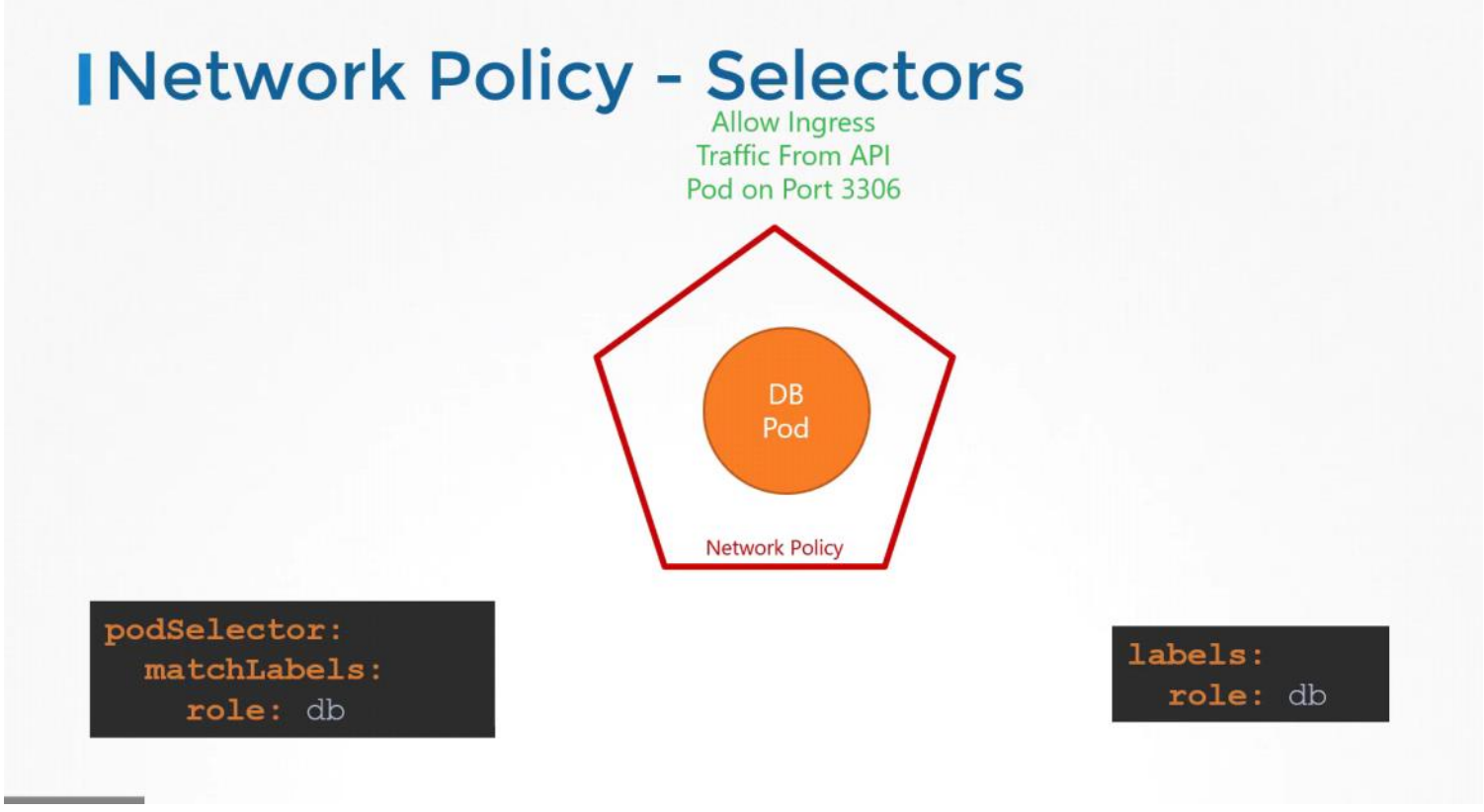




allowed traffic from any pod to pod or any other service within the cluster.



here a policy can be adapted to restrict traffic from web server to app server.



Network Policy - Rules

```yaml
policyTypes:
- Ingress
ingress:
- from:
  - podSelector:
      matchLabels:
        name: api-pod
  ports:
  - protocol: TCP
    port: 3306
```

Allow Ingress Traffic From API Pod on Port 3306

Network Policy

```yaml
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
```

```
kubectl create -f policy-definition.yaml
```

traffic from this api pod will only be allowed.

Note

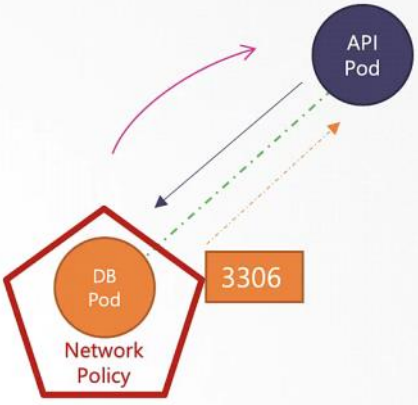
Solutions that Support Network Policies:

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

Solutions that DO NOT Support Network Policies:

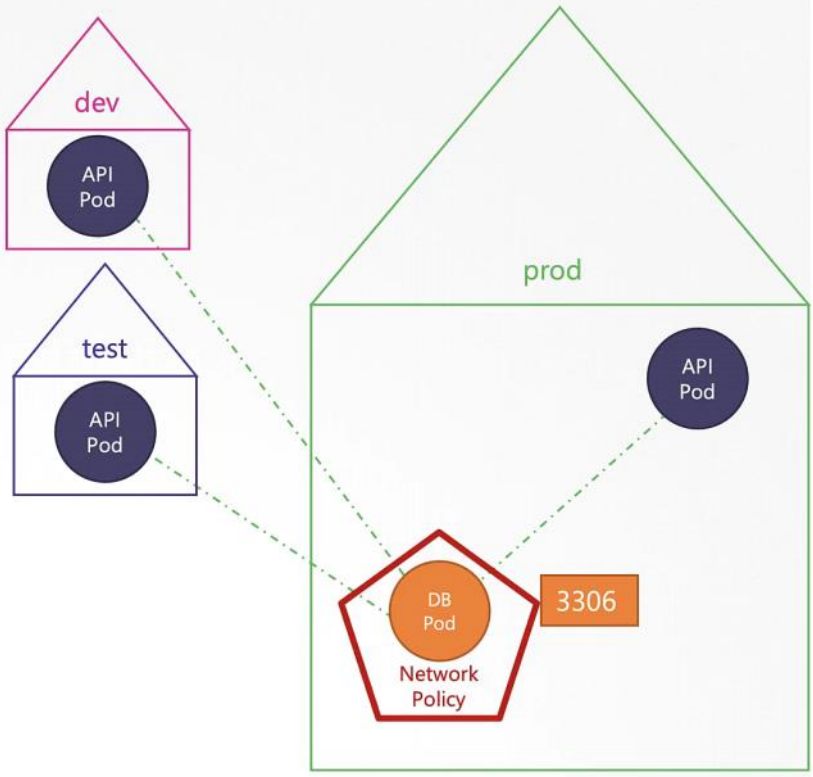
- Flannel

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: db-policy
spec:
  podSelector:
    matchLabels:
      role: db

  policyTypes:
  - Ingress
```

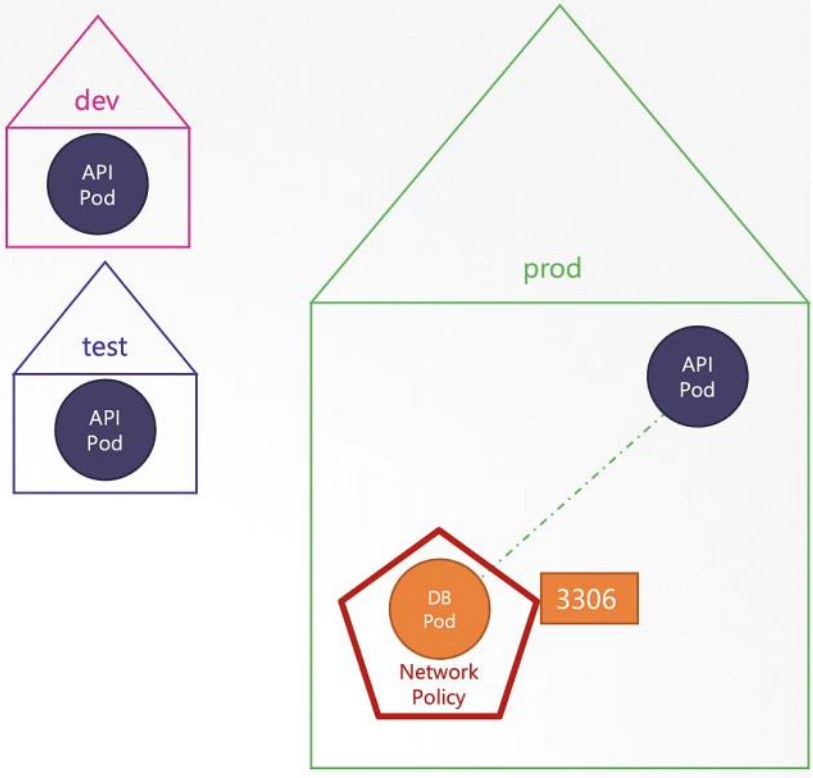


if you select ingress policy ie qweries from the api pod will be sent to db pod you don't need to set egress policy as the qwery results are automatically sent to the api pod.

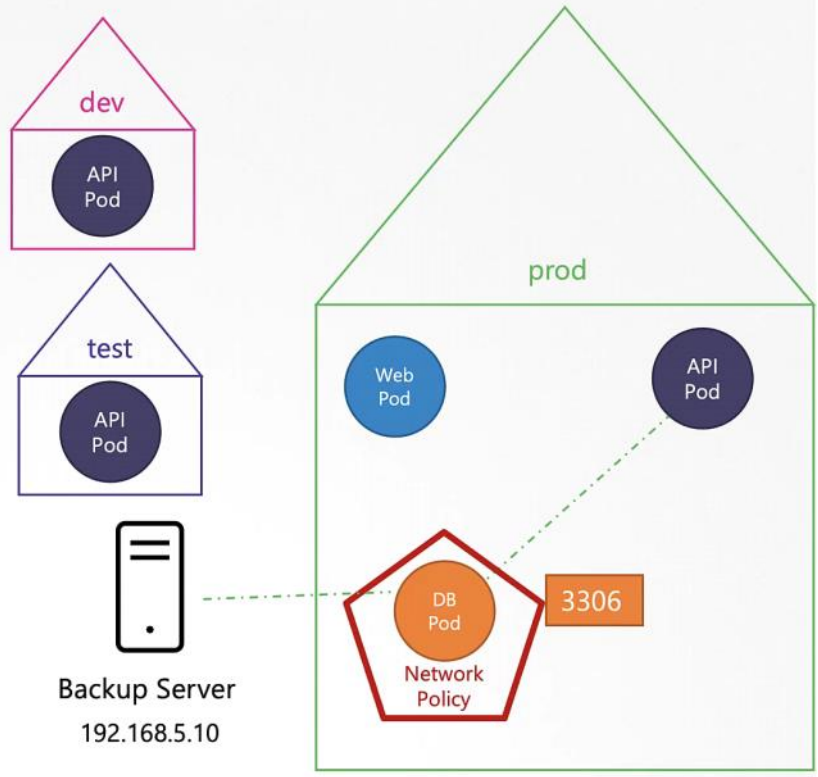
```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: db-policy
spec:
  podSelector:
    matchLabels:
      role: db

  policyTypes:
  - Ingress

  ingress:
  - from:
    - podSelector:
          matchLabels:
            name: api-pod
    ports:
    - protocol: TCP
      port: 3306
```



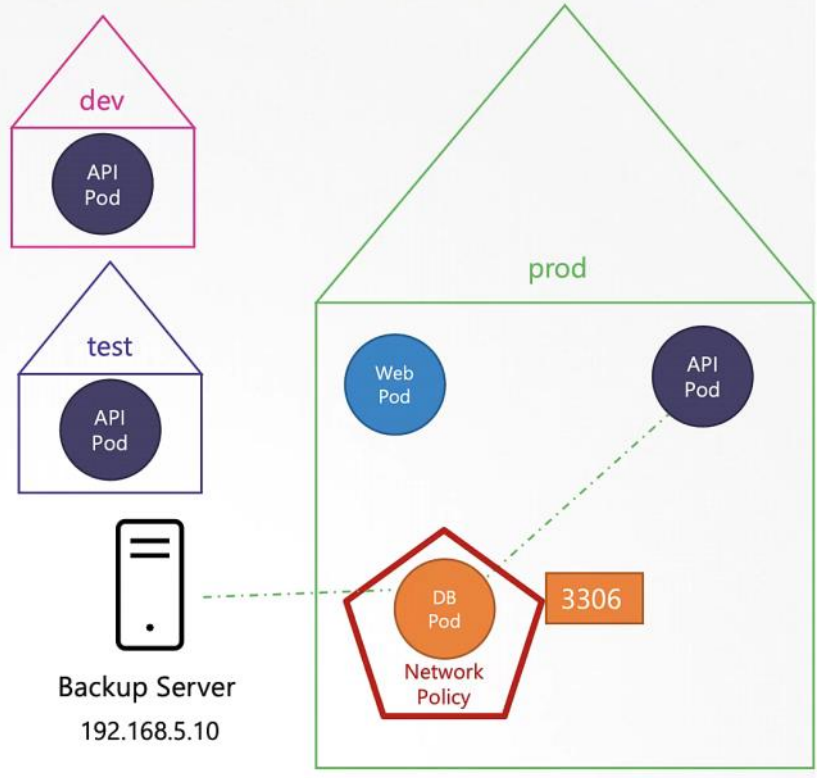

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: db-policy
spec:
  podSelector:
    matchLabels:
      role: db

  policyTypes:
  - Ingress

  ingress:
  - from:
    - podSelector:
          matchLabels:
            name: api-pod
      namespaceSelector:
          matchLabels:
            name: prod
    ports:
    - protocol: TCP
      port: 3306
```



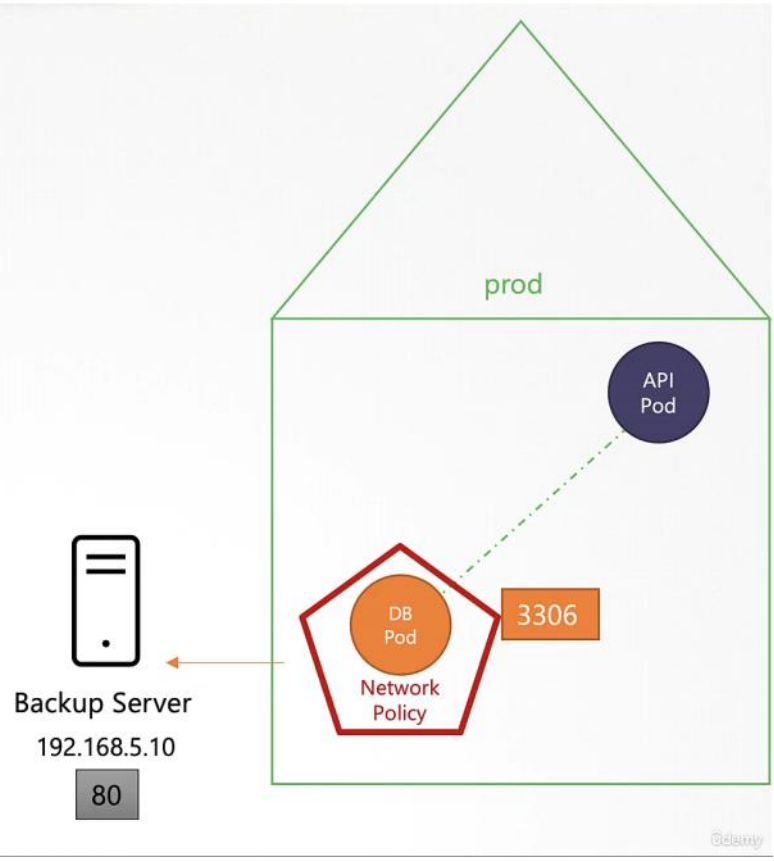
```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: db-policy
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          name: api-pod
      namespaceSelector:
        matchLabels:
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




```yaml
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
```



```yaml
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
```



Practice- network policies

Friday, July 5, 2024 9:07 AM

01 02 03 04 05 06 07 08 09 10

How many network policies do you see in the environment?

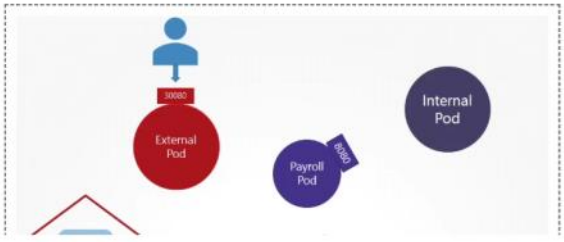
We have deployed few web applications, services and network policies. Inspect the environment.

3

4

5

1




```
root@controlplane:~#
root@controlplane:~# k get pods
NAME       READY   STATUS    RESTARTS   AGE
external   1/1     Running   0          2m20s
internal   1/1     Running   0          2m19s
mysql      1/1     Running   0          2m19s
payroll    1/1     Running   0          2m19s
root@controlplane:~#
root@controlplane:~# k get service
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
db-service         ClusterIP   10.109.89.42    <none>        3306/TCP         2m42s
external-service   NodePort    10.108.170.44   <none>        8080:30080/TCP   2m42s
internal-service   NodePort    10.98.11.243    <none>        8080:30082/TCP   2m42s
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP          39m
payroll-service    NodePort    10.110.165.31   <none>        8080:30083/TCP   2m42s
root@controlplane:~#
```

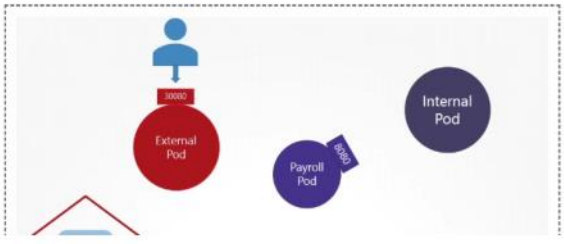
Driver Update Me

Task Hint 56:07 Terminal 1 +

01 02 03 04 05 06 07 08 09 10

How many network policies do you see in the environment?

We have deployed few web applications, services and network policies. Inspect the environment.




```
root@controlplane:~#
root@controlplane:~# k get pods
NAME       READY   STATUS    RESTARTS   AGE
external   1/1     Running   0          2m20s
internal   1/1     Running   0          2m19s
mysql      1/1     Running   0          2m19s
payroll    1/1     Running   0          2m19s
root@controlplane:~#
root@controlplane:~# k get service
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
db-service         ClusterIP   10.109.89.42    <none>        3306/TCP         2m42s
external-service   NodePort    10.108.170.44   <none>        8080:30080/TCP   2m42s
internal-service   NodePort    10.98.11.243    <none>        8080:30082/TCP   2m42s
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP          39m
payroll-service    NodePort    10.110.165.31   <none>        8080:30083/TCP   2m42s
root@controlplane:~#
root@controlplane:~# k get networkpoliciess
error: the server doesn't have a resource type "networkpoliciess"
root@controlplane:~# k get networkpolicies
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m25s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get netpol
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m31s
root@controlplane:~#
root@controlplane:~#
```

Driver Update Me

Task Hint 56:04 Terminal 1 + External Service Internal Portal

```
root@controlplane:~#
root@controlplane:~# k get pods
```

Task Hint 56:04

01 02 03 04 05 06 07 08 09 10

What is the name of the Network Policy?

network-policy

policy-1

payroll-policy

deny-policy

Driver Update Me

Terminal 1 + External Service Internal Portal

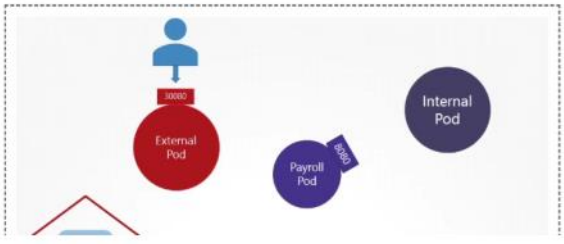
```
root@controlplane:~#
root@controlplane:~# k get pods
NAME        READY   STATUS    RESTARTS   AGE
external    1/1     Running   0          2m20s
internal    1/1     Running   0          2m19s
mysql       1/1     Running   0          2m19s
payroll     1/1     Running   0          2m19s
root@controlplane:~#
root@controlplane:~# k get service
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
db-service          ClusterIP   10.109.89.42    <none>        3306/TCP         2m42s
external-service    NodePort    10.108.170.44   <none>        8080:30080/TCP   2m42s
internal-service    NodePort    10.98.11.243    <none>        8080:30082/TCP   2m42s
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP          39m
payroll-service     NodePort    10.110.165.31   <none>        8080:30083/TCP   2m42s
root@controlplane:~#
root@controlplane:~# k get networkpoliciess
error: the server doesn't have a resource type "networkpoliciess"
root@controlplane:~# k get networkpolicies
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m25s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get netpol
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m31s
root@controlplane:~#
root@controlplane:~#
```

Task Hint 56:07

01 02 03 04 05 06 07 08 09 10

How many network policies do you see in the environment?

We have deployed few web applications, services and network policies. Inspect the environment.



Driver Update Me

Terminal 1 +

```
root@controlplane:~#
root@controlplane:~# k get pods
NAME        READY   STATUS    RESTARTS   AGE
external    1/1     Running   0          2m20s
internal    1/1     Running   0          2m19s
mysql       1/1     Running   0          2m19s
payroll     1/1     Running   0          2m19s
root@controlplane:~#
root@controlplane:~# k get service
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
db-service          ClusterIP   10.109.89.42    <none>        3306/TCP         2m42s
external-service    NodePort    10.108.170.44   <none>        8080:30080/TCP   2m42s
internal-service    NodePort    10.98.11.243    <none>        8080:30082/TCP   2m42s
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP          39m
payroll-service     NodePort    10.110.165.31   <none>        8080:30083/TCP   2m42s
root@controlplane:~#
root@controlplane:~# k get networkpoliciess
error: the server doesn't have a resource type "networkpoliciess"
root@controlplane:~# k get networkpolicies
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m25s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get netpol
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m31s
root@controlplane:~#
root@controlplane:~#
```

Task Hint Solution 55:33

01 02 03 04 05 06 07 08 09 10

Which pod is the Network Policy applied on?

- payroll
- payroll-service
- external
- internal

Terminal 1

```
root@controlplane:~#
root@controlplane:~# k get pods
NAME        READY   STATUS    RESTARTS   AGE
external    1/1     Running   0          2m20s
internal    1/1     Running   0          2m19s
mysql       1/1     Running   0          2m19s
payroll     1/1     Running   0          2m19s
root@controlplane:~#
root@controlplane:~# k get service
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
db-service          ClusterIP   10.109.89.42    <none>        3306/TCP         2m42s
external-service    NodePort    10.108.170.44   <none>        8080:30080/TCP   2m42s
internal-service    NodePort    10.98.11.243    <none>        8080:30082/TCP   2m42s
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP          39m
payroll-service     NodePort    10.110.165.31   <none>        8080:30083/TCP   2m42s
root@controlplane:~#
root@controlplane:~# k get networkpoliciess
error: the server doesn't have a resource type "networkpoliciess"
root@controlplane:~# k get networkpolicies
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m25s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get netpol
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m31s
root@controlplane:~#
root@controlplane:~#
```

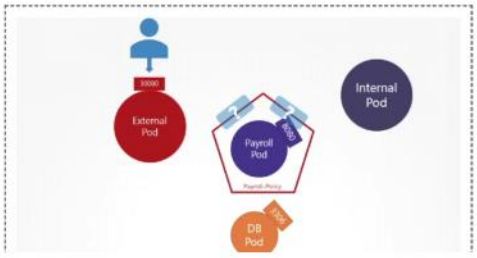
Driver Update Me

Task Hint 53:52

01 02 03 04 05 06 07 08 09 10

What type of traffic is this Network Policy configured to handle?

- Both Ingress and Egress
- Ingress
- None
- Egress



Terminal 1

```
mysql       1/1     Running   0          2m19s
payroll     1/1     Running   0          2m19s
root@controlplane:~#
root@controlplane:~# k get service
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
db-service          ClusterIP   10.109.89.42    <none>        3306/TCP         2m42s
external-service    NodePort    10.108.170.44   <none>        8080:30080/TCP   2m42s
internal-service    NodePort    10.98.11.243    <none>        8080:30082/TCP   2m42s
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP          39m
payroll-service     NodePort    10.110.165.31   <none>        8080:30083/TCP   2m42s
root@controlplane:~#
root@controlplane:~# k get networkpoliciess
error: the server doesn't have a resource type "networkpoliciess"
root@controlplane:~# k get networkpolicies
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m25s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get netpol
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m31s
root@controlplane:~#
root@controlplane:~# k describe netpol payroll-policy
Name:         payroll-policy
Namespace:    default
Created on:   2022-04-18 20:35:54 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     name=payroll
  Allowing ingress traffic:
    To Port: 8080/TCP
    From:
      PodSelector: name=internal
  Not affecting egress traffic
  Policy Types: Ingress
root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

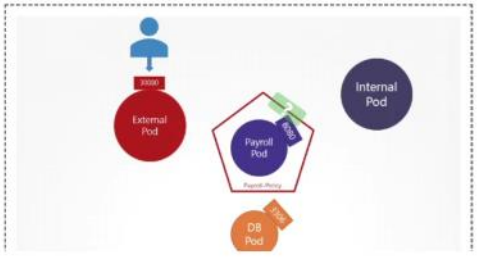
Driver Update Me

Task 53:23

01 02 03 04 05 06 07 08 09 10

What is the impact of the rule configured on this Network Policy?

- Traffic To and From Payroll POD is Blocked
- Traffic From Internal to Payroll POD is blocked
- Traffic To and From Internal POD is Blocked
- Traffic From Internal to Payroll POD is allowed



Terminal 1

```
mysql       1/1     Running   0          2m19s
payroll     1/1     Running   0          2m19s
root@controlplane:~#
root@controlplane:~# k get service
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
db-service         ClusterIP   10.109.89.42    <none>        3306/TCP         2m42s
external-service   NodePort    10.108.170.44   <none>        8080:30080/TCP   2m42s
internal-service   NodePort    10.98.11.243    <none>        8080:30082/TCP   2m42s
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP          39m
payroll-service    NodePort    10.110.165.31   <none>        8080:30083/TCP   2m42s
root@controlplane:~#
root@controlplane:~# k get networkpoliciess
error: the server doesn't have a resource type "networkpoliciess"
root@controlplane:~# k get networkpolicies
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m25s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get netpol
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m31s
root@controlplane:~#
root@controlplane:~# k describe netpol payroll-policy
Name:         payroll-policy
Namespace:    default
Created on:   2022-04-18 20:35:54 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     name=payroll
  Allowing ingress traffic:
    To Port: 8080/TCP
    From:
      PodSelector: name=internal
  Not affecting egress traffic
  Policy Types: Ingress
root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

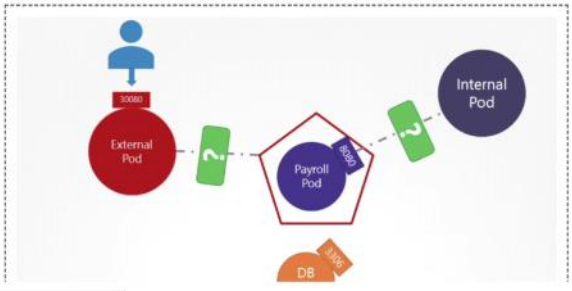
Driver Update Me

Task 50:42

01 02 03 04 05 06 07 08 09 10

Perform a connectivity test using the User Interface in these Applications to access the `payroll-service` at port `8080`.

- Both internal and external applications can access payroll service
- Only internal application can access payroll service



Terminal 1 External Portal Internal Portal

```
s
internal-service   NodePort    10.98.11.243    <none>        8080:30082/TCP   2m42
s
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP          39m
payroll-service    NodePort    10.110.165.31   <none>        8080:30083/TCP   2m42
s
root@controlplane:~#
root@controlplane:~# k get networkpoliciess
error: the server doesn't have a resource type "networkpoliciess"
root@controlplane:~# k get networkpolicies
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m25s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get netpol
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m31s
root@controlplane:~#
root@controlplane:~# k describe netpol payroll-policy
Name:         payroll-policy
Namespace:    default
Created on:   2022-04-18 20:35:54 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     name=payroll
  Allowing ingress traffic:
    To Port: 8080/TCP
    From:
      PodSelector: name=internal
  Not affecting egress traffic
  Policy Types: Ingress
root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

82x35

Driver Update Me

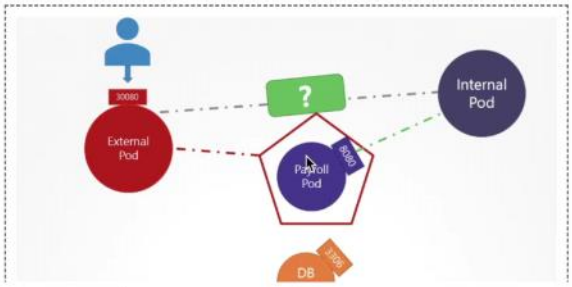
Task 50:00

01 02 03 04 05 06 07 08 09 10

Perform a connectivity test using the User Interface of the Internal Application to access the `external-service` at port `8080`.

Successful

Failed



Terminal 1 + External Portal Internal Portal

```
s
internal-service   NodePort    10.98.11.243    <none>        8080:30082/TCP   2m42
s
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP          39m
payroll-service    NodePort    10.110.165.31   <none>        8080:30083/TCP   2m42
s
root@controlplane:~#
root@controlplane:~# k get networkpoliciess
error: the server doesn't have a resource type "networkpoliciess"
root@controlplane:~# k get networkpolicies
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m25s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k get netpol
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   3m31s
root@controlplane:~#
root@controlplane:~# k describe netpol payroll-policy
Name:         payroll-policy
Namespace:    default
Created on:   2022-04-18 20:35:54 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     name=payroll
  Allowing ingress traffic:
    To Port: 8080/TCP
    From:
      PodSelector: name=internal
  Not affecting egress traffic
  Policy Types: Ingress
root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

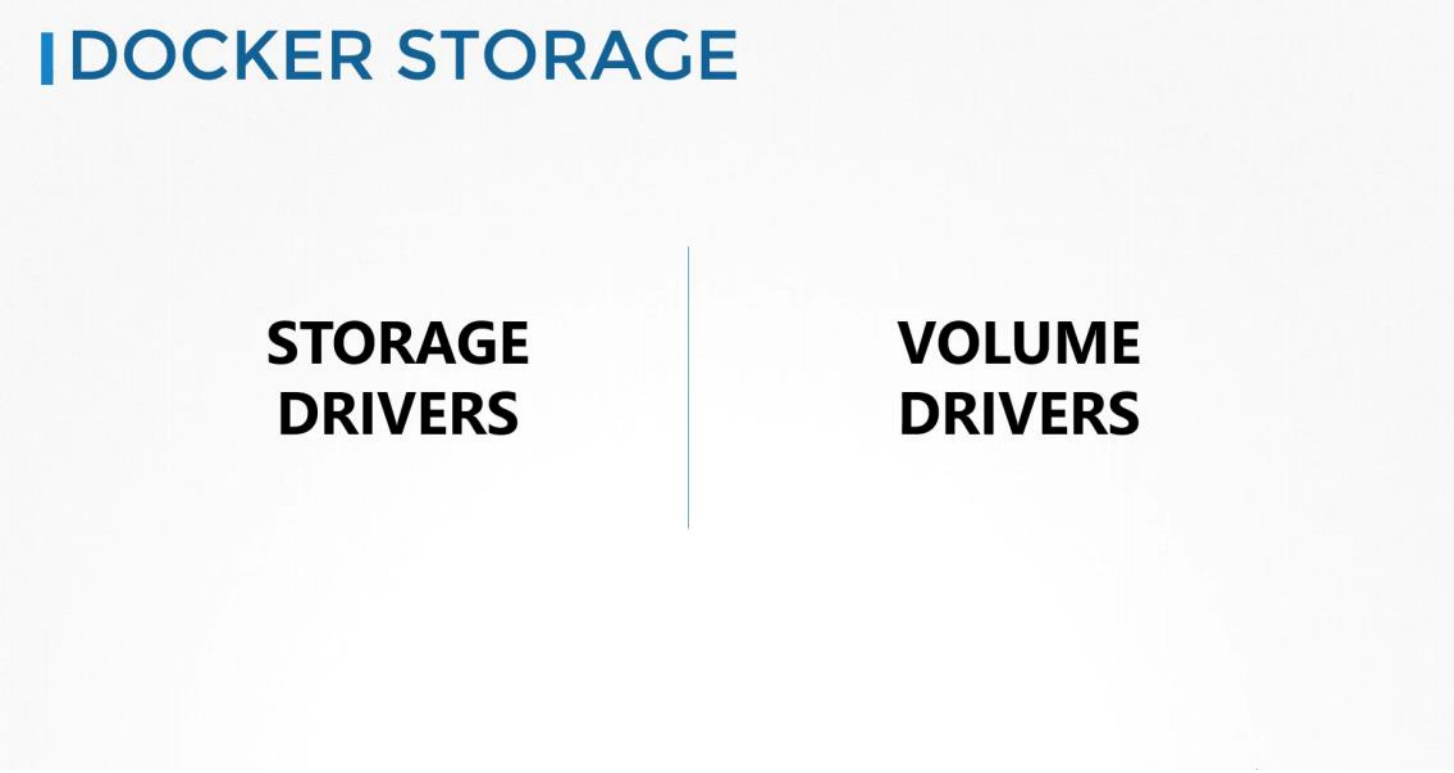
```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# vi internal-policy.yaml
```

```yaml
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

~
~
~
```

```
  PodSelector:     name=payroll
  Allowing ingress traffic:
    To Port: 8080/TCP
    From:
      PodSelector: name=internal
  Not affecting egress traffic
  Policy Types: Ingress
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# vi internal-policy.yaml
root@controlplane:~# vi internal-policy.yaml
root@controlplane:~#
root@controlplane:~# k create -f internal-policy.yaml
networkpolicy.networking.k8s.io/internal-policy created
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# k describe netpol internal-policy
Name:         internal-policy
Namespace:    default
Created on:   2022-04-18 20:53:13 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     name=internal
  Not affecting ingress traffic
  Allowing egress traffic:
    To Port: 8080/TCP
    To:
      PodSelector: name=payroll
    ----------
    To Port: 3306/TCP
    To:
      PodSelector: name=mysql
  Policy Types: Egress
root@controlplane:~#
```

Storage

Monday, July 8, 2024 10:44 AM



Layered architecture

Dockerfile

```
FROM Ubuntu

RUN apt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask
run
```

```
docker build Dockerfile -t mmumshad/my-custom-app
```

Layer 1. Base Ubuntu Layer	120 MB
Layer 2. Changes in apt packages	306 MB
Layer 3. Changes in pip packages	6.3 MB
Layer 4. Source code	229 B
Layer 5. Update Entrypoint	0 B

Layered architecture

Dockerfile

```
FROM Ubuntu

RUN apt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask
run
```

```
docker build Dockerfile -t mmumshad/my-custom-app
```

Layer 1. Base Ubuntu Layer	120 MB
Layer 2. Changes in apt packages	306 MB
Layer 3. Changes in pip packages	6.3 MB
Layer 4. Source code	229 B
Layer 5. Update Entrypoint	0 B

Dockerfile2

```
FROM Ubuntu

RUN apt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY app2.py /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app2.py flask
run
```

```
docker build Dockerfile2 -t mmumshad/my-custom-app-2
```

- Layer 1. Base Ubuntu Layer
- Layer 2. Changes in apt packages
- Layer 3. Changes in pip packages
- Layer 4. Source code
- Layer 5. Update Entrypoint

river Update Me
udemy

Layered architecture

Container Layer

Read Write

Layer 6. Container Layer

```
docker run mmumshad/my-custom-app
```

Image Layers

Read Only

Layer 5. Update Entrypoint with "flask" command

Layer 4. Source code

Layer 3. Changes in pip packages

Layer 2. Changes in apt packages

Layer 1. Base Ubuntu Layer

```
docker build Dockerfile -t mmumshad/my-custom-app
```

volumes

```
docker run -v data_volume:/var/lib/mysql mysql
```

```
docker volume create data_volume
```

/var/lib/docker

volumes

data_volume

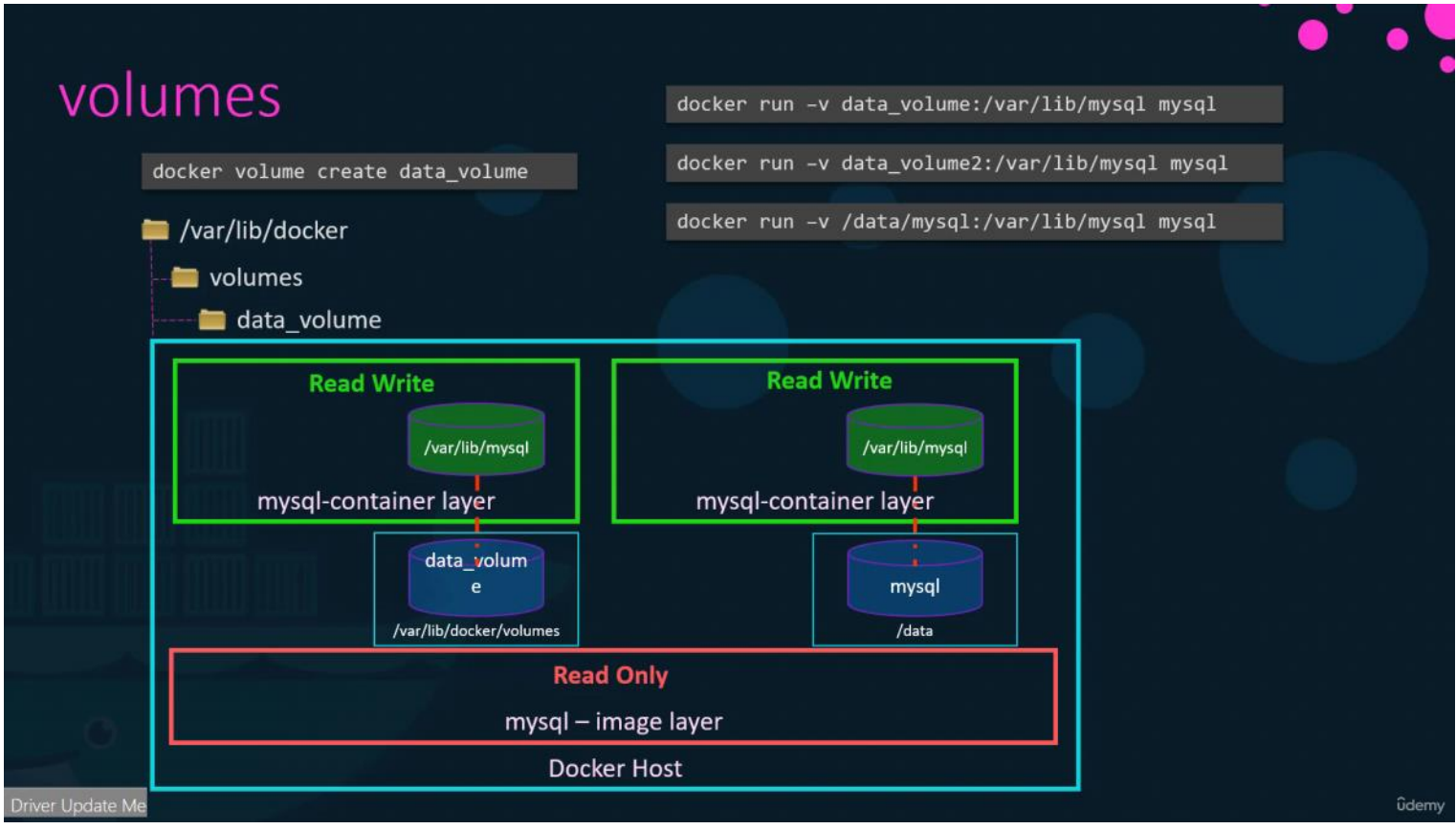
data_volum
e

/var/lib/docker/volumes

Read Only

mysql – image layer

Docker Host




```
docker run -v data_volume:/var/lib/mysql mysql

docker run -v data_volume2:/var/lib/mysql mysql

docker run -v /data/mysql:/var/lib/mysql mysql

docker run \
--mount type=bind,source=/data/mysql,target=/var/lib/mysql mysql
```

Storage drivers

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

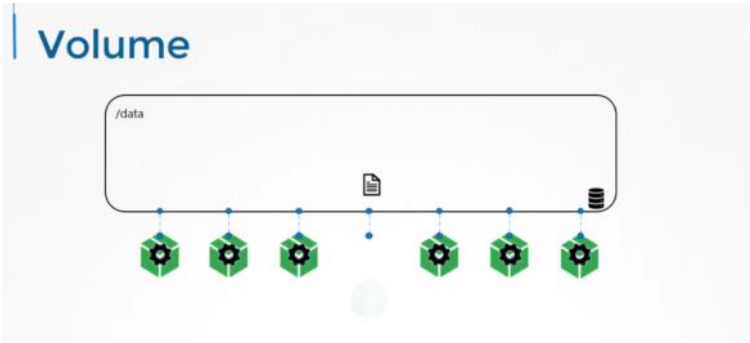
ubuntu=AUFS

Volumes

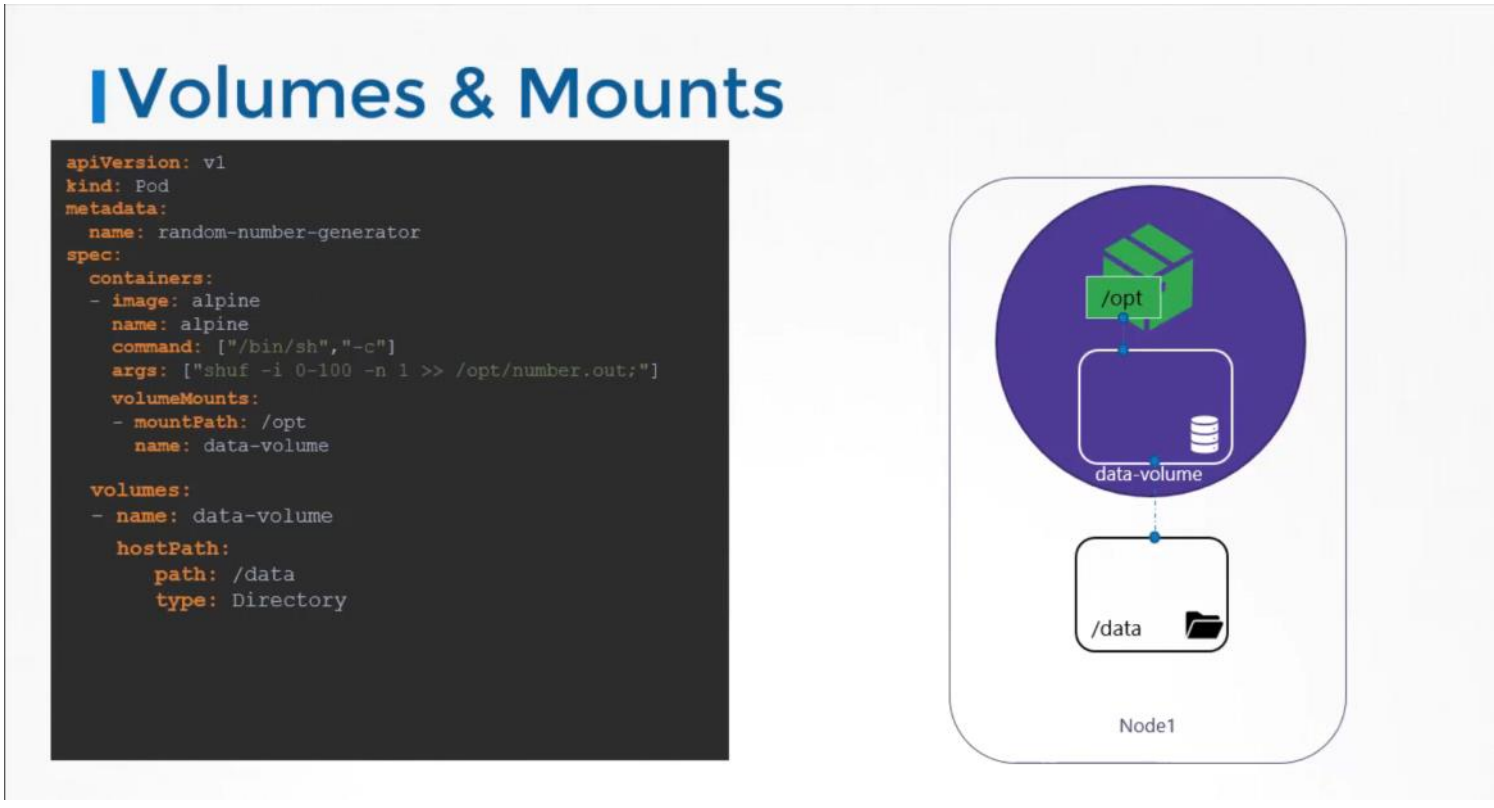
Thursday, August 22, 2024 5:18 PM

There are two types of mounts

1. volume mount: directly mounts volume
2. bind mount: binds a directory to a container



data processed by then container is placed in this volume thereby retainin it permanently even if the datat is deleted



Volume Types

```
volumes:
- name: data-volume
  awsElasticBlockStore:
    volumeID: <volume-id>
    fsType: ext4
```

```
volumes:
- name: data-volume
  awsElasticBlockStore:
    volumeID: <volume-id>
    fsType: ext4
```

Persistent Volume

pv-definition.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
   name: pv-vol1
spec:
  accessModes:
      - ReadWriteOnce
  capacity:
      storage: 1Gi
  awsElasticBlockStore:
    volumeID: <volume-id>
    fsType: ext4
```

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

Persistent Volume (PV)

```
kubectl get persistentvolume
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
pv-vol1	1Gi	RWO	Retain	Available				3m

Persistent volume claims

Saturday, August 24, 2024 11:42 AM

Persistent Volume Claim

pvc-definition.yaml

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

```
kubectl get persistentvolumeclaim
NAME        STATUS    VOLUME    CAPACITY    ACCESS MODES
myclaim     Pending
```

```
kubectl create -f pvc-definition.yaml
```

We see the claim in a pending state.

Persistent Volume Claim

pvc-definition.yaml

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

```
kubectl create -f pvc-definition.yaml
```

pv-definition.yaml

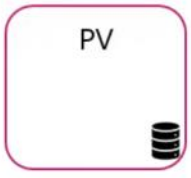
```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-vol1
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  awsElasticBlockStore:
    volumeID: <volume-id>
    fsType: ext4
```

View PVCs

```
kubectl get persistentvolumeclaim
NAME        STATUS    VOLUME    CAPACITY   ACCESS MODES   STORAGECLASS   AGE
myclaim     Bound     pv-vol1   1Gi        RWO                           43m
```

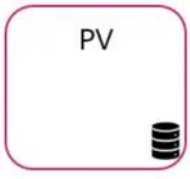
Delete PVCs

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



Delete PVCs

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




```
persistentVolumeReclaimPolicy:  Recycle
```

Using PVCs in Pods
Once you create a PVC use it in a POD definition file by specifying the PVC Claim

name under persistentVolumeClaim section in the volumes section like this:

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim
```

The same is true for ReplicaSets or Deployments. Add this to the pod template section of a Deployment on ReplicaSet.

Practice Test - Persistent Volume Claims

Saturday, August 24, 2024 12:14 PM

Next →

2 / 19

The application stores logs at location `/log/app.log`. View the logs.

You can exec in to the container and open the file:

```
kubectl exec webapp -- cat /log/app.log
```

Ok

```
he item is OUT OF STOCK.
[2024-08-24 06:44:22,830] INFO in event-simulator: USER4 is viewing page1
[2024-08-24 06:44:23,831] INFO in event-simulator: USER4 logged out
[2024-08-24 06:44:24,832] INFO in event-simulator: USER3 is viewing page3
[2024-08-24 06:44:25,833] INFO in event-simulator: USER3 is viewing page3
[2024-08-24 06:44:26,834] WARNING in event-simulator: USER5 Failed to Login a
s the account is locked due to MANY FAILED ATTEMPTS.
[2024-08-24 06:44:26,835] INFO in event-simulator: USER2 is viewing page3
[2024-08-24 06:44:27,836] INFO in event-simulator: USER3 logged out
[2024-08-24 06:44:28,837] INFO in event-simulator: USER2 logged in
[2024-08-24 06:44:29,838] INFO in event-simulator: USER3 logged out
[2024-08-24 06:44:30,839] WARNING in event-simulator: USER7 Order failed as t
he item is OUT OF STOCK.
[2024-08-24 06:44:30,839] INFO in event-simulator: USER3 logged in
[2024-08-24 06:44:31,840] WARNING in event-simulator: USER5 Failed to Login a
s the account is locked due to MANY FAILED ATTEMPTS.
[2024-08-24 06:44:31,841] INFO in event-simulator: USER2 is viewing page2
```

kubectl exec webapp -- cat /log/app.log

Task Hint Solution 55:30

Next →

3 / 19

If the POD was to get deleted now, would you be able to view these logs.

No

Yes

Terminal 1

```
controlplane ~ ➜ k get po
No resources found in default namespace.

controlplane ~ ➜ k exec webapp -- less /log/app.log
Error from server (NotFound): pods "webapp" not found

controlplane ~ ✗
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Configure a volume to store these logs at `/var/log/webapp` on the host.

Use the spec provided below.

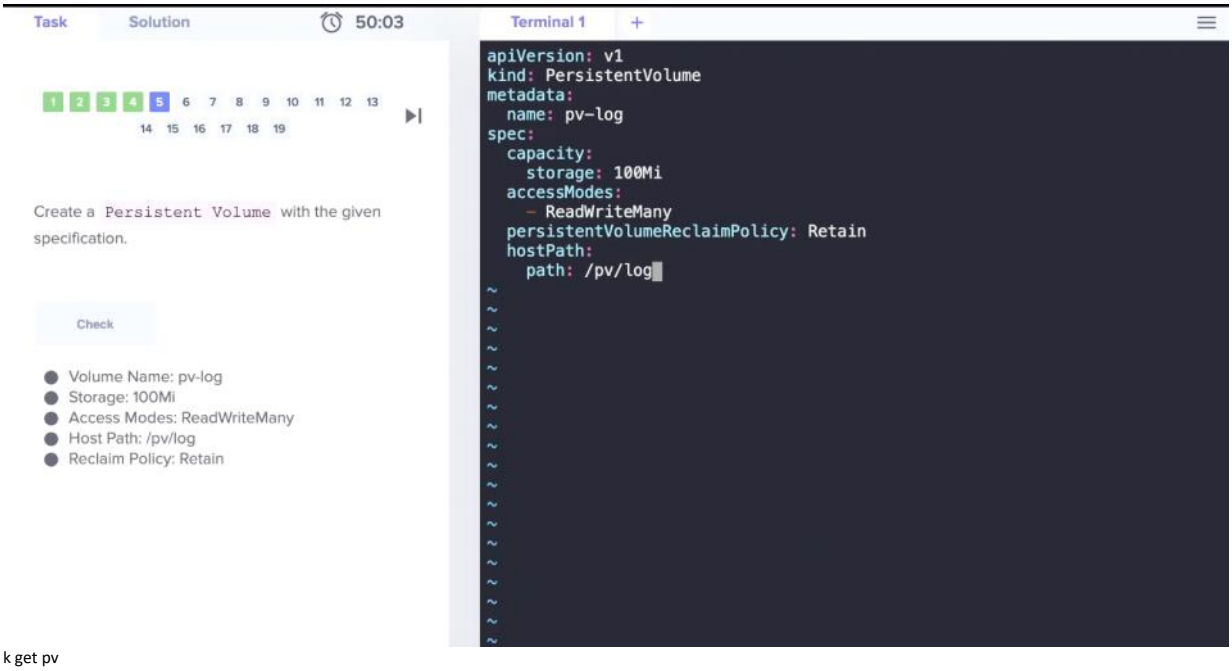
Check

- Name: webapp
- Image Name: kodekloud/event-simulator
- Volume HostPath: /var/log/webapp
- Volume Mount: /log

```yaml
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-lmntb
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: controlplane
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  volumes:
  - name: log-volume
    hostPath:
      path: /var/log/webapp
  - name: kube-api-access-lmntb
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          expirationSeconds: 3607
          path: token
      - configMap:
          items:
          - key: ca.crt
            path: ca.crt
```

k edit po webapp

```
    image: kodekloud/event-simulator
    imagePullPolicy: Always
    name: event-simulator
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-lmntb
      readOnly: true
    - mountPath: /log
      name: log-volume
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: controlplane
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  volumes:
  - name: log-volume
    hostPath:
      path: /var/log/webapp
  - name: kube-api-access-lmntb
    projected:
      defaultMode: 420
```



k get pv

Practice -storage class

Saturday, August 24, 2024 11:54 AM

Next →

1 / 13

How many `StorageClasses` exist in the cluster right now?

- 3
- 7
- 5
- 4

```
All rights reserved

controlplane ~ ➜ k get sc
NAME                   PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGM
ODE      ALLOWVOLUMEEXPANSION   AGE
local-path (default)   rancher.io/local-path   Delete          WaitForFirstCo
nsumer   false                  19m

controlplane ~ ➜ k get sc | wc -l
2
```

Next →

3 / 13

What is the name of the `Storage Class` that does not support `dynamic` volume provisioning?

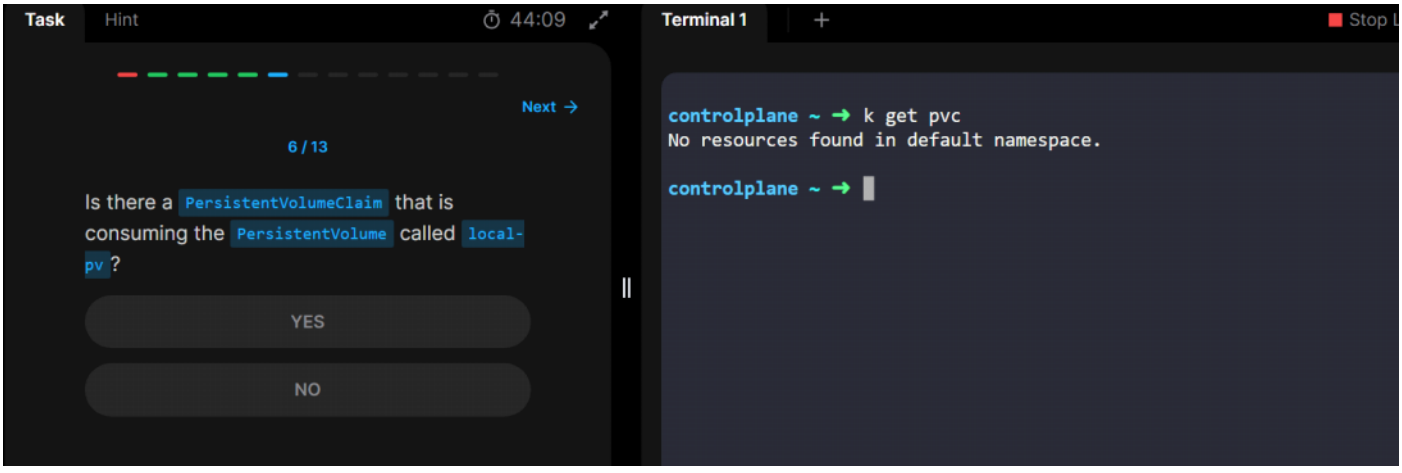
- portWorx-Storageclass
- nfs-sc
- local-storage
- glusterfs-sc

```
controlplane ~ ➜ k get sc --no-headers | wc -l
3

controlplane ~ ➜ k get sc
NAME                      PROVISIONER                    RECLAIMPOLICY   V
OLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
local-path (default)      rancher.io/local-path          Delete          W
aitForFirstConsumer   false                  23m
local-storage             kubernetes.io/no-provisioner   Delete          W
aitForFirstConsumer   false                  3m31s
portworx-io-priority-high   kubernetes.io/portworx-volume   Delete          I
mmediate              false                  3m31s

controlplane ~ ➜
```

```
controlplane ~ ➜ k get sc --no-headers | wc -l
3

controlplane ~ ➜ k get sc
NAME                      PROVISIONER                    RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
local-path (default)      rancher.io/local-path          Delete          WaitForFirstConsumer   false                  23m
local-storage             kubernetes.io/no-provisioner   Delete          WaitForFirstConsumer   false                  3m31s
portworx-io-priority-high kubernetes.io/portworx-volume   Delete          Immediate              false                  3m31s

controlplane ~ ➜
```

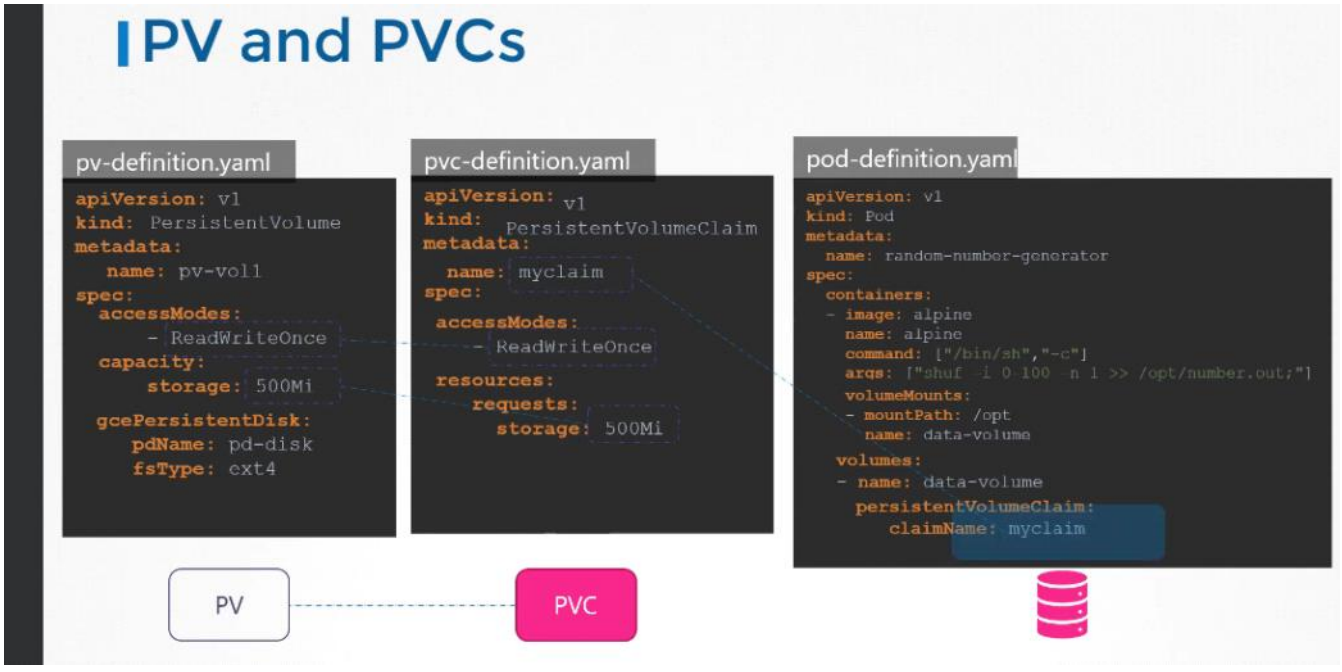
Look for the storage class name that uses no-provisioner

```
controlplane ~ ➜  k get sc
NAME                    PROVISIONER                    RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
local-path (default)    rancher.io/local-path          Delete          WaitForFirstConsumer   false
23m
local-storage           kubernetes.io/no-provisioner   Delete          WaitForFirstConsumer   false
3m31s
portworx-io-priority-high   kubernetes.io/portworx-volume   Delete          Immediate           false
3m31s
```



Storage Class

Saturday, August 24, 2024 2:20 PM



Storage Class

sc-definition.yaml

```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.yam

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

sc-platinum-definition.yam

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: platinum
provisioner: kubernetes.io/gce-p
parameters:
  type: pd-ssd
  replication-type: regional-pd
```

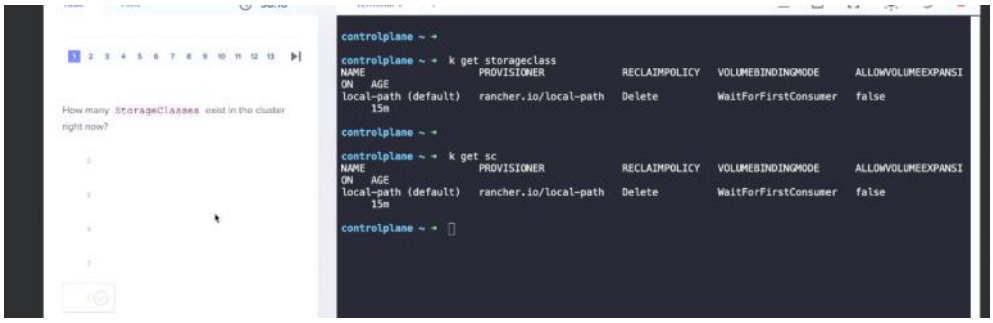
Silver SC

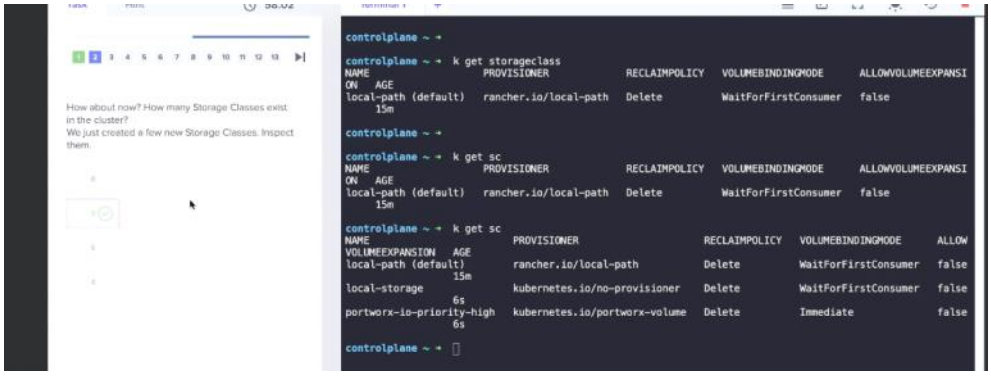
Gold SC

Platinum SC

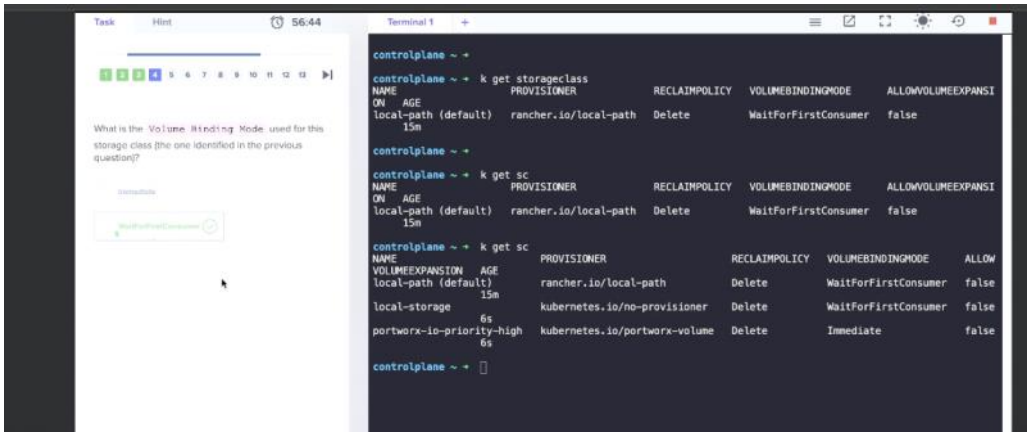
Practice -storage class

Saturday, August 24, 2024 2:24 PM





volume that does not support provsisioning = local storage



What is the `Provisioner` used for the storage class called `portworx-io-priority-high` ?

```
controlplane ~ ➜

controlplane ~ ➜  k get storageclass
NAME                 PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSI
ON   AGE
local-path (default)   rancher.io/local-path   Delete          WaitForFirstConsumer   false
     15m

controlplane ~ ➜

controlplane ~ ➜  k get sc
NAME                 PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSI
ON   AGE
local-path (default)   rancher.io/local-path   Delete          WaitForFirstConsumer   false
     15m

controlplane ~ ➜  k get sc
NAME                   PROVISIONER                     RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOW
VOLUMEEXPANSION   AGE
local-path (default)   rancher.io/local-path           Delete          WaitForFirstConsumer   false
                  15m
local-storage          kubernetes.io/no-provisioner    Delete          WaitForFirstConsumer   false
                  6s
portworx-io-priority-high   kubernetes.io/portworx-volume   Delete          Immediate              false
                  6s

controlplane ~ ➜
```

Is there a `PersistentVolumeClaim` that is consuming the `PersistentVolume` called `local-pv` ?

```
controlplane ~ ➜  k get pv
NAME       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS    REASON   AGE
local-pv   500Mi      RWO            Retain           Available           local-storage            110s

controlplane ~ ➜  k get pvc
No resources found in default namespace.

controlplane ~ ➜
```

Task Hint Solution 53:42 Terminal 1

Let's fix that. Create a new `PersistentVolumeClaim` by the name of `local-pvc` that should bind to the volume `local-pv`.

Inspect the pv `local-pv` for the specs.

```
controlplane ~ ➜  k get pv
NAME       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS    REASON   AGE
local-pv   500Mi      RWO            Retain           Available           local-storage            110s

controlplane ~ ➜  k get pvc
No resources found in default namespace.

controlplane ~ ➜  vi pvc.yaml

controlplane ~ ➜  k create -f pvc.yaml
persistentvolumeclaim/local-pvc created

controlplane ~ ➜

controlplane ~ ➜  k get pvc
NAME        STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS    AGE
local-pvc   Pending                                      local-storage   4s

controlplane ~ ➜
```

Why is the PVC in a pending state despite making a valid request to claim the volume called `local-pv` ?

Inspect the PVC events.

- Storage Class not found
- PVC Specs are incorrect
- A Pod consuming the volume is not scheduled
- Persistent Volume and Claim mismatch

```
controlplane ~ ➜  k get pv
NAME       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS    REASON   AGE
local-pv   500Mi      RWO            Retain           Available           local-storage            110s

controlplane ~ ➜  k get pvc
No resources found in default namespace.

controlplane ~ ➜  vi pvc.yaml

controlplane ~ ➜  k create -f pvc.yaml
persistentvolumeclaim/local-pvc created

controlplane ~ ➜

controlplane ~ ➜  k get pvc
NAME        STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS    AGE
local-pvc   Pending                                      local-storage   4s

controlplane ~ ➜  k describe pvc local-pvc
Name:          local-pvc
Namespace:     default
StorageClass:  local-storage
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Used By:       <none>
Events:
  Type    Reason                Age                From                         Message
  ----    ------                ----               ----                         -------
  Normal  WaitForFirstConsumer  6s (x4 over 49s)   persistentvolume-controller  waiting for first consume
r to be created before binding

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13

Create a new pod called `nginx` with the image `nginx:alpine`. The Pod should make use of the PVC `local-pvc` and mount the volume at the path `/var/www/html`.

The PV `local-pv` should in a bound state.

Check

- Pod created with the correct Image?
- Pod uses PVC called local-pvc?
- local-pv bound?
- nginx pod running?
- Volume mounted at the correct path?

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx:alpine
    name: nginx
    resources: {}
    volumeMounts:
      - mountPath: "/var/www/html"
        name: local-pvc-volume
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
  - name: local-pvc-volume
    persistentVolumeClaim:
        claimName: local-pvc
status: {}
```

the name of the volume mount in container will be the volume name ie created below.

summary

Tuesday, August 20, 2024 1:09 PM

kubectl taint node controlplane node-role.kubernetes.io/control-plane:NoSchedule-

diff bw taint and untaint command is the - arg given at the end

kubectl label node node01 color=blue --> label a node

kubectl edit deployment blue

kubectl get pods -o wide

kubectl get po -A

kubectl create -f . ---> creates all files

kubectl top node

kubectl top pod

kubectl logs -f podname

kubectl edit deployment frontend

kubectl create configmap --from-literal=APP_COLOR=darkblue

kubectl get secrets

kubectl create secret generic db-secret --from-literal=DB_Hos
t=sql01 --from-literal=DB_User=root --from-literal=DB_Password=password123
secret/db-secret created

echo "secretname" | base64 --decode

cd /etc/kubernetes/pki/etcd/ --> has all the cert files

ps -aux | grep kube-api

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

kubectl drain node01 --ignore-daemonsets

What is the current version of the cluster?
k get nodes

openssl x509 -in /etc/kubernetes/pki/apiserver.crt -text

k get csr
cat rohan.csr | base64 -w 0

k certificate deny rohan --> reject the request
k delete csr rohan --> delete the csr request
k approve certificate rohan --> approve the cert req

k get roles -A

kubectl describe rolebinding kube-proxy -n kube-system

k get pods --as dev-user

k auth can-i get pods --as dev-user --> check user permissions

kubectl create rolebinding dev-user-binding --namespace=default --role=developer --user=dev-user
kubectl create role developer --namespace=default --verb=list,create,delete --resource=pod

cluster roles are cluster wide and not part of any namespace

k api-resources --> get api groups and resources

k get sa --> get service account

k create secret docker-registry

kubectl create secret docker-registry private-reg-cred --docker-username=dock_user --docker-password=dock_password --docker-server=myprivateregistry.com:5000 --docker-email=dock_user@myprivateregistry.co

k get networkpolicies
k get netpol

k get pvc
k get pv

kubectl exec webapp -- cat /log/app.log

A quick note on editing Pods and Deployments

Edit a POD
Remember, you CANNOT edit specifications of an existing POD other than the below.

- spec.containers[*].image
- spec.initContainers[*].image
- spec.activeDeadlineSeconds
- spec.tolerations

For example you cannot edit the environment variables, service accounts, resource limits (all of which we will discuss later) of a running pod. But if you really want to, you have 2 options:
1. Run the kubectl edit pod <pod name> command. This will open the pod

specification in an editor (vi editor). Then edit the required properties. When you try to save it, you will be denied. This is because you are attempting to edit a field on the pod that is not editable.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
# pods "webapp" was not valid:
# * spec: Forbidden: pod updates may not change fields other than `spec.containers[*].image`,
# {"Volumes":[{"Name":"default-token-xcr8k","HostPath":null,"EmptyDir":null,"GCEPersistentDis
:null},"NFS":null,"ISCSI":null,"Glusterfs":null,"PersistentVolumeClaim":null,"RBD":null,"Quob
phereVolume":null,"AzureDisk":null,"PhotonPersistentDisk":null,"Projected":null,"PortworxVolu
":null,"WorkingDir":"","Ports":null,"EnvFrom":null,"Env":null,"Resources":{"Limits":null,"Req
ubPath":"","MountPropagation":null}],"VolumeDevices":null,"LivenessProbe":null,"ReadinessProb
curityContext":null,"Stdin":false,"StdinOnce":false,"TTY":false},{"Name":"agentx","Image":"ag
meMounts":[{"Name":"default-token-xcr8k","ReadOnly":true,"MountPath":"/var/run/secrets/kubern
":null,"TerminationMessagePath":"/dev/termination-log","TerminationMessagePolicy":"File","Ima
iodSeconds":30,"ActiveDeadlineSeconds":null,"DNSPolicy":"ClusterFirst","NodeSelector":null,"S
#
# A: 2","AutomountServiceAccountToken":null,"NodeName":"node01","SecurityContext":{"HostNetwo
Root":null,"SupplementalGroups":null,"FSGroup":null,"Sysctls":null},"ImagePullSecrets":null,"
rator":"Exists","Value":"","Effect":"NoExecute","TolerationSeconds":300},{"Key":"node.kuberne
","Priority":0,"DNSConfig":null,"ReadinessGates":null}
#
# B: ","AutomountServiceAccountToken":null,"NodeName":"node01","SecurityContext":{"HostNetwor
oot":null,"SupplementalGroups":null,"FSGroup":null,"Sysctls":null},"ImagePullSecrets":null,"H
ator":"Exists","Value":"","Effect":"NoExecute","TolerationSeconds":300},{"Key":"node.kubernet
","Priority":0,"DNSConfig":null,"ReadinessGates":null}
#


#
apiVersion: v1
"/tmp/kubectl-edit-ccvrq.yaml" 125L, 7010C
```

```
master $ kubectl edit pod webapp
error: pods "webapp" is invalid
A copy of your changes has been stored to "/tmp/kubectl-edit-ccvrq.yaml"
error: Edit cancelled, no valid changes were saved.
master $
```

A copy of the file with your changes is saved in a temporary location as shown above. You can then delete the existing pod by running the command:

kubectl delete pod webapp

Then create a new pod with your changes using the temporary file

kubectl create -f /tmp/kubectl-edit-ccvrq.yaml

2. The second option is to extract the pod definition in YAML format to a file using the command

kubectl get pod webapp -o yaml > my-new-pod.yaml

Then make the changes to the exported file using an editor (vi editor). Save the changes

vi my-new-pod.yaml

Then delete the existing pod

kubectl delete pod webapp

Then create a new pod with the edited file

kubectl create -f my-new-pod.yaml

Edit Deployments

With Deployments you can easily edit any field/property of the POD template. Since the pod template is a child of the deployment specification, with every change the

deployment will automatically delete and create a new pod with the new changes. So if you are asked to edit a property of a POD part of a deployment you may do that simply by running the command
kubectl edit deployment my-deployment

1. Init containers

Init containers in Kubernetes are specialized containers that run before the main application containers in a Pod. They are designed to perform setup or initialization tasks that must be completed before the main application containers start. Here's a detailed overview:

Purpose:

- **Initialization Tasks**: They can perform tasks such as setting up configuration files, waiting for a service to be available, or initializing databases.
- **Dependency Management**: They ensure that all dependencies are met before the main application starts.

Example Use Cases

- **Database Initialization**: Running a script to initialize a database schema before the application starts.
- **Configuration Setup**: Fetching configuration files from a remote server and placing them in a shared volume.
- **Waiting for Services**: Ensuring that a dependent service is up and running before starting the main application.

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
  - name: myapp-container
    image: myapp:latest
    ports:
    - containerPort: 80
  initContainers:
  - name: init-myservice
    image: busybox
    command: ['sh', '-c', 'until nslookup myservice; do echo waiting for myservice; sleep 2; done;']
  - name: init-db
    image: busybox
    command: ['sh', '-c', 'until nslookup mydb; do echo waiting for mydb; sleep 2; done;']
```

2. Using Readiness Probes

Readiness probes can be used to ensure that a container is ready to serve traffic. You can configure a readiness probe to check the availability of a dependent resource.

```
spec:
  containers:
  - name: myapp-container
    image: myapp
    readinessProbe:
      httpGet:
        path: /healthz
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 10
```

1. **liveness probe**

Using a liveness probe in Kubernetes helps ensure that your application is running smoothly by periodically checking the health of your containers. If a container fails the liveness probe, Kubernetes will restart it. Here's how you can set up a liveness probe:

Steps to Configure a Liveness Probe
Define the Liveness Probe in Your Pod Specification: You can define a liveness probe using one of three methods: HTTP request, TCP socket, or an exec command. Here's an example using each method:
HTTP Request:

```
apiVersion: v1
kind: Pod
metadata:
  name: liveness-http
spec:
  containers:
  - name: myapp
    image: myapp:latest
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8080
      initialDelaySeconds: 3
      periodSeconds: 3
```

Readiness Probe

Purpose: To determine if a container is ready to start accepting traffic. If the readiness probe fails, the container is removed from the service load balancers.

Use Case: Ensuring that a container is fully initialized and ready to serve requests, such as after loading configuration files or establishing network connections.

```
readinessProbe:
  httpGet:
```

```
    path: /ready
    port: 8080
  initialDelaySeconds: 5
  periodSeconds: 5
```

3. Startup Probe

Purpose: To determine if an application within a container has started. This is particularly useful for applications that take a long time to start.

Use Case: Preventing Kubernetes from killing the container before it has fully started. If a startup probe is configured, it disables liveness and readiness checks until it succeeds.

```
startupProbe:
  httpGet:
    path: /startup
    port: 8080
  initialDelaySeconds: 10
  periodSeconds: 10
  failureThreshold: 30
```