MOCK EXAM 1.0

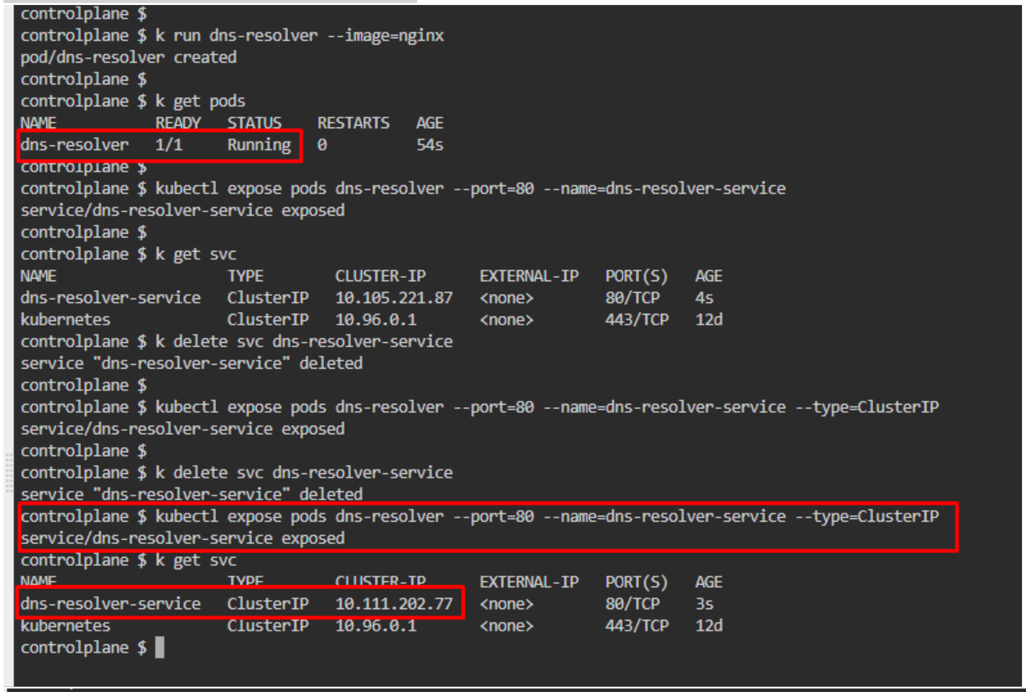
Name: Haffiz Hissham

Date: 22 September 2024

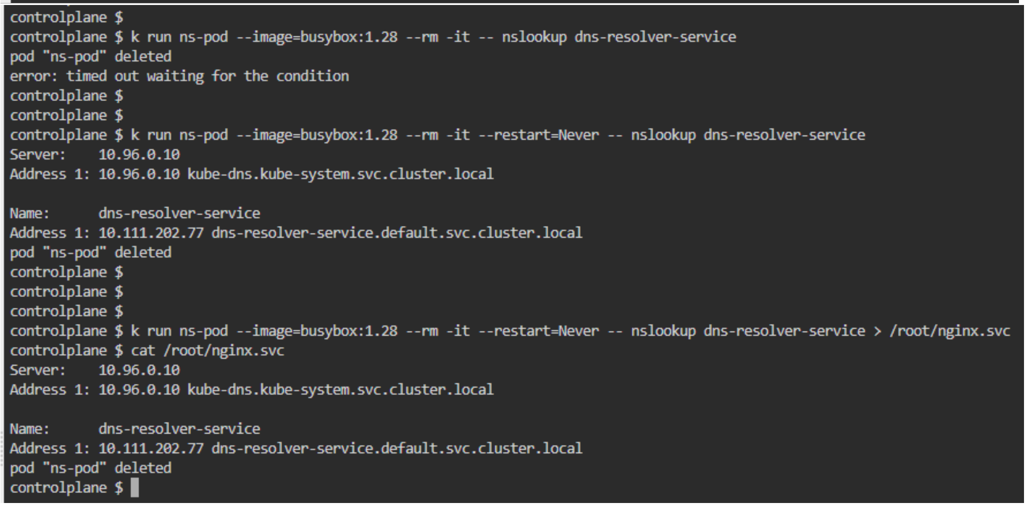
Q1

Create a nginx pod called dns-resolver using image nginx expose it internally with a service called dns-resolver-service.

check if pod and service name are resolvable from within the cluster.
use the image: busybox:1.28 for dns lookup
save the result in /root/nginx.svc.



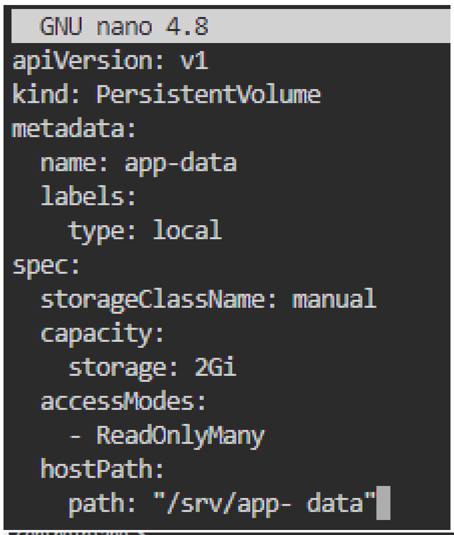
```
controlplane $
controlplane $ k run dns-resolver --image=nginx
pod/dns-resolver created
controlplane $
controlplane $ k get pods
NAME           READY   STATUS    RESTARTS   AGE
dns-resolver   1/1     Running   0          54s
controlplane $
controlplane $ kubectl expose pods dns-resolver --port=80 --name=dns-resolver-service
service/dns-resolver-service exposed
controlplane $
controlplane $ k get svc
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
dns-resolver-service   ClusterIP   10.105.221.87   <none>        80/TCP    4s
kubernetes             ClusterIP   10.96.0.1       <none>        443/TCP   12d
controlplane $ k delete svc dns-resolver-service
service "dns-resolver-service" deleted
controlplane $
controlplane $ kubectl expose pods dns-resolver --port=80 --name=dns-resolver-service --type=ClusterIP
service/dns-resolver-service exposed
controlplane $
controlplane $ k delete svc dns-resolver-service
service "dns-resolver-service" deleted
controlplane $ kubectl expose pods dns-resolver --port=80 --name=dns-resolver-service --type=ClusterIP
service/dns-resolver-service exposed
controlplane $ k get svc
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
dns-resolver-service   ClusterIP   10.111.202.77   <none>        80/TCP    3s
kubernetes             ClusterIP   10.96.0.1       <none>        443/TCP   12d
controlplane $
```



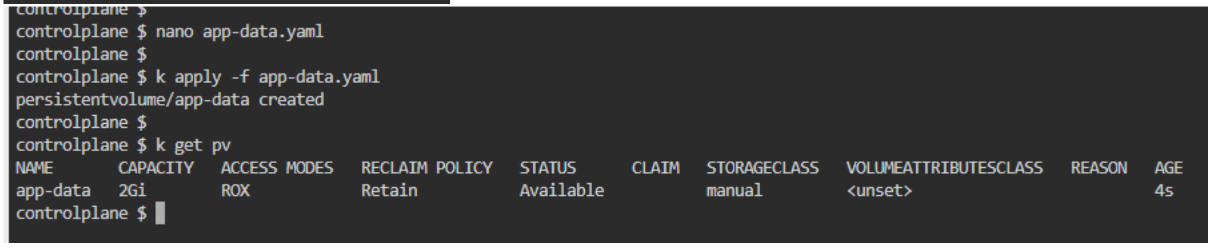
```
controlplane $
controlplane $ k run ns-pod --image=busybox:1.28 --rm -it -- nslookup dns-resolver-service
pod "ns-pod" deleted
error: timed out waiting for the condition
controlplane $
controlplane $
controlplane $ k run ns-pod --image=busybox:1.28 --rm -it --restart=Never -- nslookup dns-resolver-service
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      dns-resolver-service
Address 1: 10.111.202.77 dns-resolver-service.default.svc.cluster.local
pod "ns-pod" deleted
controlplane $
controlplane $
controlplane $
controlplane $ k run ns-pod --image=busybox:1.28 --rm -it --restart=Never -- nslookup dns-resolver-service > /root/nginx.svc
controlplane $ cat /root/nginx.svc
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      dns-resolver-service
Address 1: 10.111.202.77 dns-resolver-service.default.svc.cluster.local
pod "ns-pod" deleted
controlplane $
```

Q2

Create a persistent volume with name app-data, of capacity 2Gi and access mode ReadOnlyMany.
The type of volume is hostPath and its location is /srv/app- data



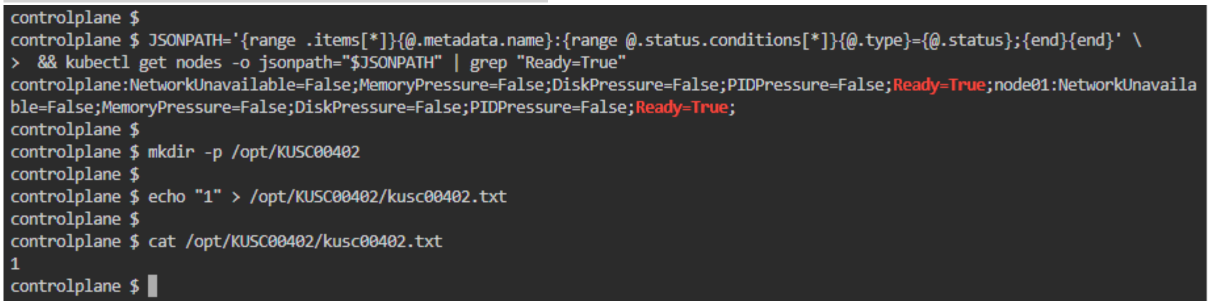
```
  GNU nano 4.8
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-data
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 2Gi
  accessModes:
    - ReadOnlyMany
  hostPath:
    path: "/srv/app- data"
```



```
controlplane $
controlplane $ nano app-data.yaml
controlplane $
controlplane $ k apply -f app-data.yaml
persistentvolume/app-data created
controlplane $
controlplane $ k get pv
NAME       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
app-data   2Gi        ROX            Retain           Available           manual         <unset>                          4s
controlplane $
```

Q3

Check to see how many nodes are ready (not including nodes tainted NoSchedule) and write the
number to /opt/KUSC00402/kusc00402.txt.

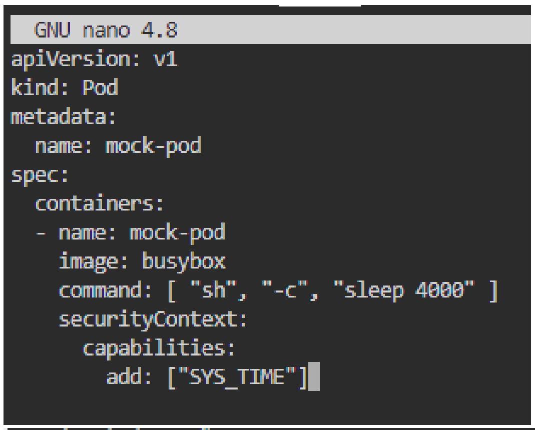


```
controlplane $
controlplane $ JSONPATH='{range .items[*]}{@.metadata.name}:{range @.status.conditions[*]}{@.type}={@.status};{end}{end}' \
>  && kubectl get nodes -o jsonpath="$JSONPATH" | grep "Ready=True"
controlplane:NetworkUnavailable=False;MemoryPressure=False;DiskPressure=False;PIDPressure=False;Ready=True;node01:NetworkUnavaila
ble=False;MemoryPressure=False;DiskPressure=False;PIDPressure=False;Ready=True;
controlplane $
controlplane $ mkdir -p /opt/KUSC00402
controlplane $
controlplane $ echo "1" > /opt/KUSC00402/kusc00402.txt
controlplane $
controlplane $ cat /opt/KUSC00402/kusc00402.txt
1
controlplane $
```

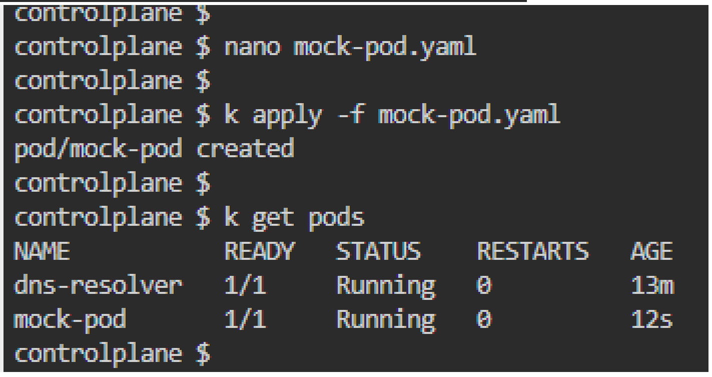
Q4

Create a new pod called mock-pod with image busy box
Allow the pod to be able to set system_time

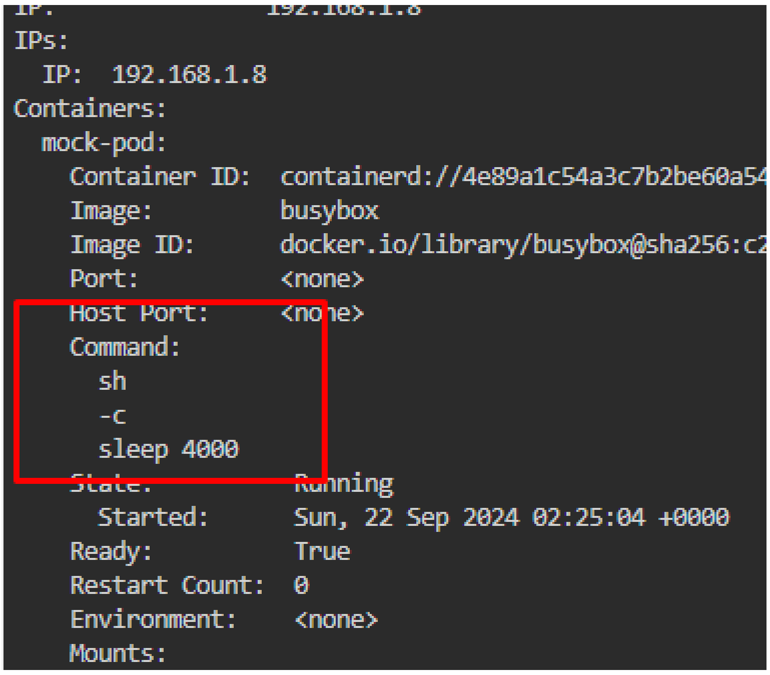
The container should sleep for 4000 seconds



```
  GNU nano 4.8
apiVersion: v1
kind: Pod
metadata:
  name: mock-pod
spec:
  containers:
  - name: mock-pod
    image: busybox
    command: [ "sh", "-c", "sleep 4000" ]
    securityContext:
      capabilities:
        add: ["SYS_TIME"]
```



```
controlplane $
controlplane $ nano mock-pod.yaml
controlplane $
controlplane $ k apply -f mock-pod.yaml
pod/mock-pod created
controlplane $
controlplane $ k get pods
NAME           READY   STATUS    RESTARTS   AGE
dns-resolver   1/1     Running   0          13m
mock-pod       1/1     Running   0          12s
controlplane $
```



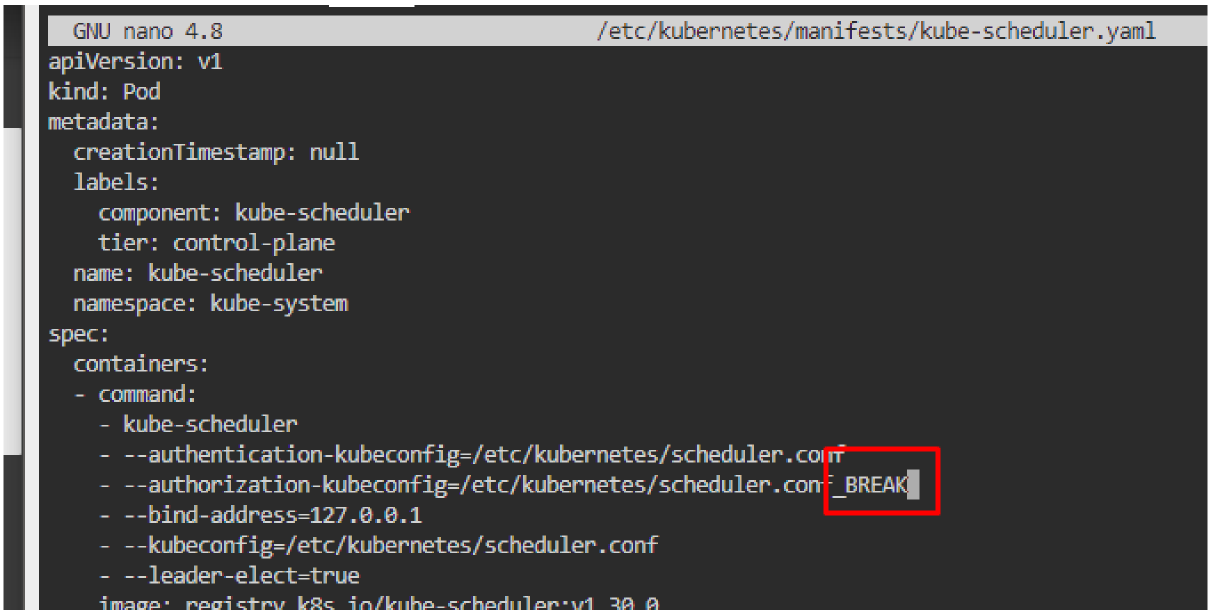
```
IP:                192.168.1.8
IPs:
  IP:  192.168.1.8
Containers:
  mock-pod:
    Container ID:  containerd://4e89a1c54a3c7b2be60a54
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:c2
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
      sleep 4000
    State:          Running
      Started:      Sun, 22 Sep 2024 02:25:04 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
```

Q5

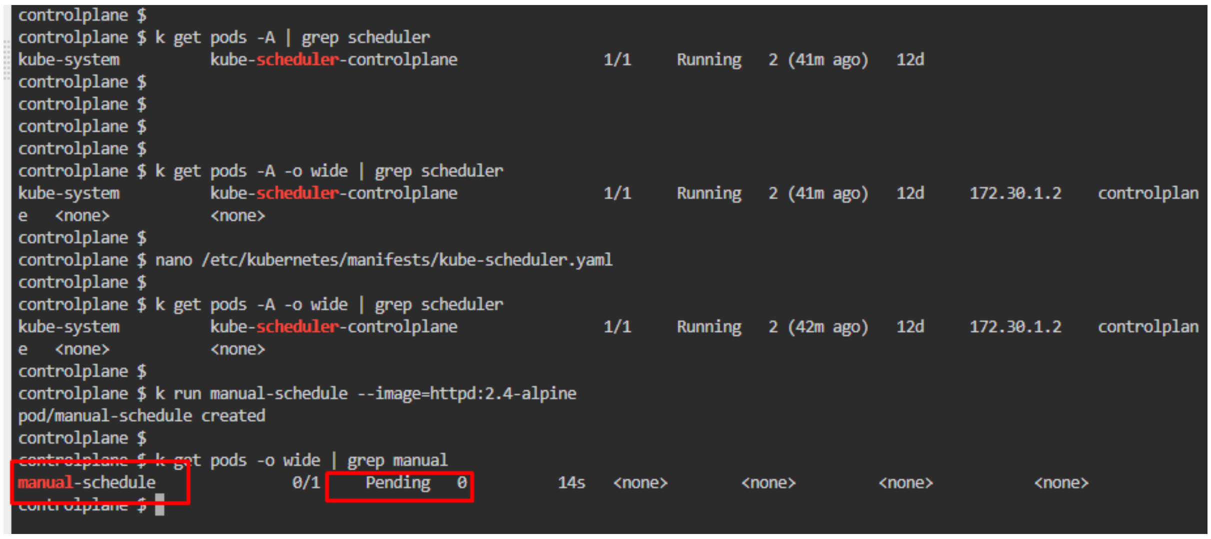
Temporarily stop the kube-scheduler, this means in a way that you can start it again afterwards.

Create a single Pod named manual-schedule of image httpd:2.4-alpine, confirm it's created but not scheduled on any node.

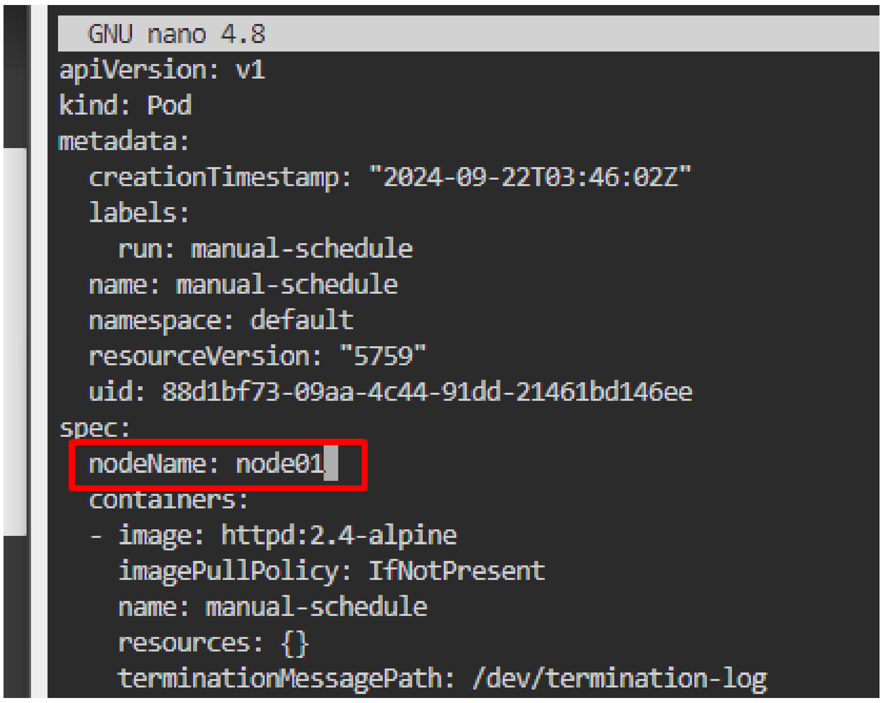
Now you're the scheduler and have all its power, manually schedule that Pod on node with nodename. Make sure it's running.
Start the kube-scheduler again and confirm it's running correctly by creating a second Pod named manual-schedule2 of image httpd:2.4-alpine on controlplane



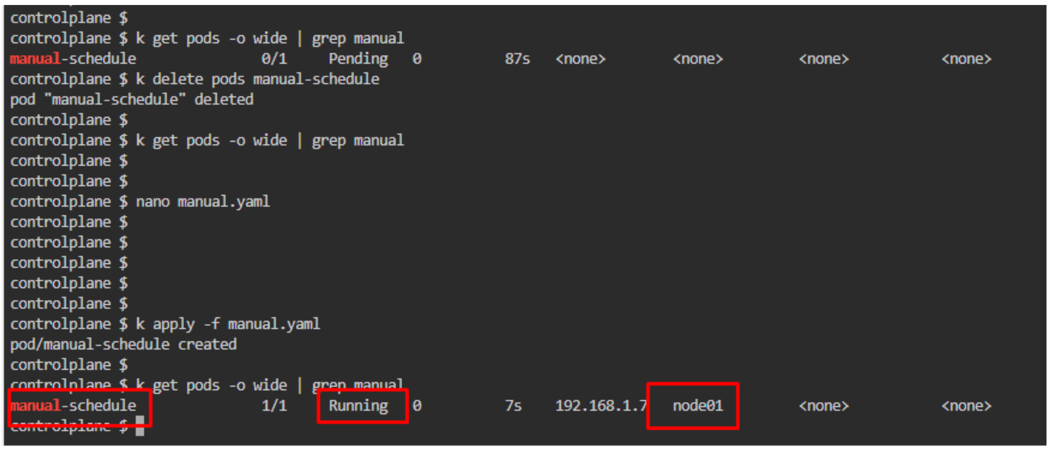
```
  GNU nano 4.8                              /etc/kubernetes/manifests/kube-scheduler.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    component: kube-scheduler
    tier: control-plane
  name: kube-scheduler
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-scheduler
    - --authentication-kubeconfig=/etc/kubernetes/scheduler.conf
    - --authorization-kubeconfig=/etc/kubernetes/scheduler.conf_BREAK
    - --bind-address=127.0.0.1
    - --kubeconfig=/etc/kubernetes/scheduler.conf
    - --leader-elect=true
    image: registry.k8s.io/kube-scheduler:v1.30.0
```



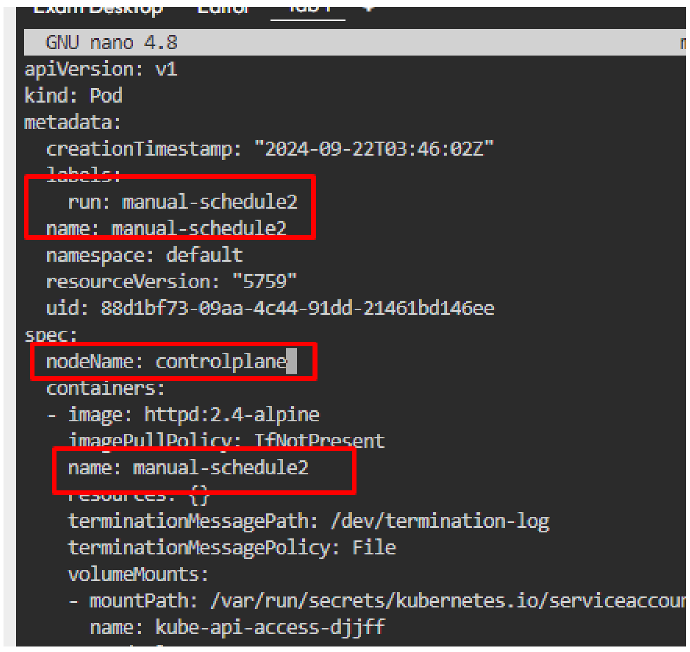
```
controlplane $
controlplane $ k get pods -A | grep scheduler
kube-system          kube-scheduler-controlplane              1/1     Running   2 (41m ago)   12d
controlplane $
controlplane $
controlplane $
controlplane $
controlplane $ k get pods -A -o wide | grep scheduler
kube-system          kube-scheduler-controlplane              1/1     Running   2 (41m ago)   12d     172.30.1.2     controlplan
e   <none>           <none>
controlplane $
controlplane $ nano /etc/kubernetes/manifests/kube-scheduler.yaml
controlplane $
controlplane $ k get pods -A -o wide | grep scheduler
kube-system          kube-scheduler-controlplane              1/1     Running   2 (42m ago)   12d     172.30.1.2     controlplan
e   <none>           <none>
controlplane $
controlplane $ k run manual-schedule --image=httpd:2.4-alpine
pod/manual-schedule created
controlplane $
controlplane $ k get pods -o wide | grep manual
manual-schedule              0/1     Pending   0          14s   <none>        <none>         <none>            <none>
controlplane $
```



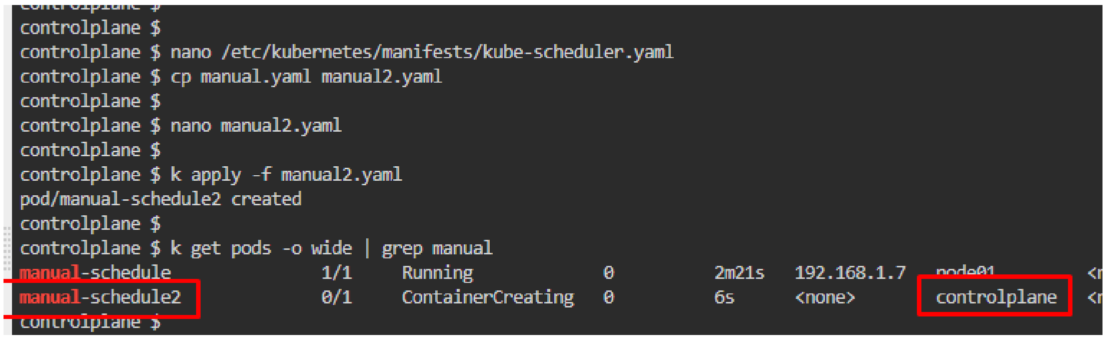
```
  GNU nano 4.8
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2024-09-22T03:46:02Z"
  labels:
    run: manual-schedule
  name: manual-schedule
  namespace: default
  resourceVersion: "5759"
  uid: 88d1bf73-09aa-4c44-91dd-21461bd146ee
spec:
  nodeName: node01
  containers:
  - image: httpd:2.4-alpine
    imagePullPolicy: IfNotPresent
    name: manual-schedule
    resources: {}
    terminationMessagePath: /dev/termination-log
```



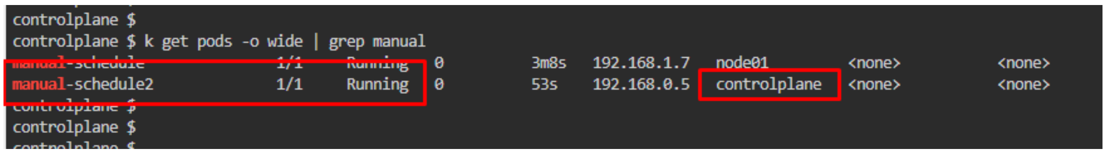
```
controlplane $
controlplane $ k get pods -o wide | grep manual
manual-schedule                0/1     Pending   0          87s   <none>        <none>        <none>           <none>
controlplane $ k delete pods manual-schedule
pod "manual-schedule" deleted
controlplane $
controlplane $ k get pods -o wide | grep manual
controlplane $
controlplane $
controlplane $ nano manual.yaml
controlplane $
controlplane $
controlplane $
controlplane $
controlplane $
controlplane $ k apply -f manual.yaml
pod/manual-schedule created
controlplane $
controlplane $ k get pods -o wide | grep manual
manual-schedule                1/1     Running   0          7s    192.168.1.7   node01        <none>           <none>
controlplane $
```



```
  GNU nano 4.8
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2024-09-22T03:46:02Z"
  labels:
    run: manual-schedule2
  name: manual-schedule2
  namespace: default
  resourceVersion: "5759"
  uid: 88d1bf73-09aa-4c44-91dd-21461bd146ee
spec:
  nodeName: controlplane
  containers:
  - image: httpd:2.4-alpine
    imagePullPolicy: IfNotPresent
    name: manual-schedule2
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccou
      name: kube-api-access-djjff
```



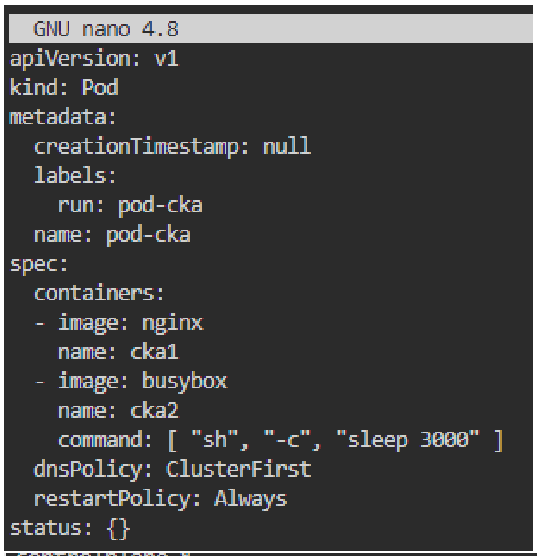
```
controlplane $
controlplane $ nano /etc/kubernetes/manifests/kube-scheduler.yaml
controlplane $ cp manual.yaml manual2.yaml
controlplane $
controlplane $ nano manual2.yaml
controlplane $
controlplane $ k apply -f manual2.yaml
pod/manual-schedule2 created
controlplane $
controlplane $ k get pods -o wide | grep manual
manual-schedule            1/1     Running             0          2m21s   192.168.1.7   node01         <r
manual-schedule2           0/1     ContainerCreating   0          6s      <none>        controlplane   <r
controlplane $
```



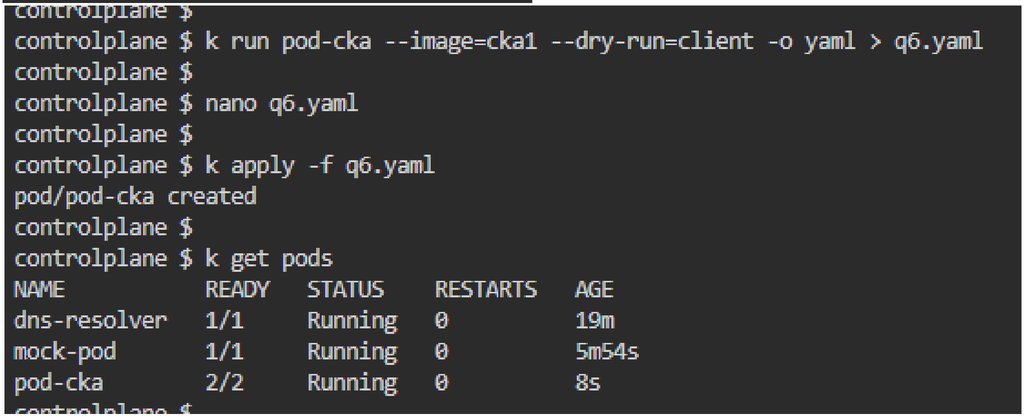
```
controlplane $
controlplane $ k get pods -o wide | grep manual
manual-schedule            1/1     Running   0          3m8s    192.168.1.7   node01         <none>           <none>
manual-schedule2           1/1     Running   0          53s     192.168.0.5   controlplane   <none>           <none>
controlplane $
controlplane $
```

Q6

Create a pod called pod-cka with two containers, as given below:
Container 1 - name: cka2, image: nginx
Container2 - name: cka2, image:
busybox,
command: sleep 3000



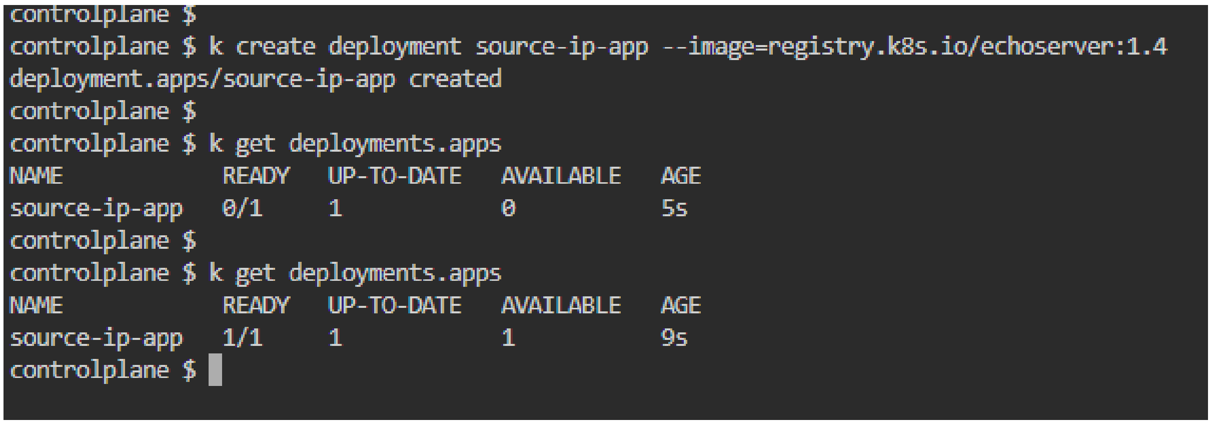
```
  GNU nano 4.8
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: pod-cka
  name: pod-cka
spec:
  containers:
  - image: nginx
    name: cka1
  - image: busybox
    name: cka2
    command: [ "sh", "-c", "sleep 3000" ]
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```



```
controlplane $
controlplane $ k run pod-cka --image=cka1 --dry-run=client -o yaml > q6.yaml
controlplane $
controlplane $ nano q6.yaml
controlplane $
controlplane $ k apply -f q6.yaml
pod/pod-cka created
controlplane $
controlplane $ k get pods
NAME           READY   STATUS    RESTARTS   AGE
dns-resolver   1/1     Running   0          19m
mock-pod       1/1     Running   0          5m54s
pod-cka        2/2     Running   0          8s
controlplane $
```

Q7

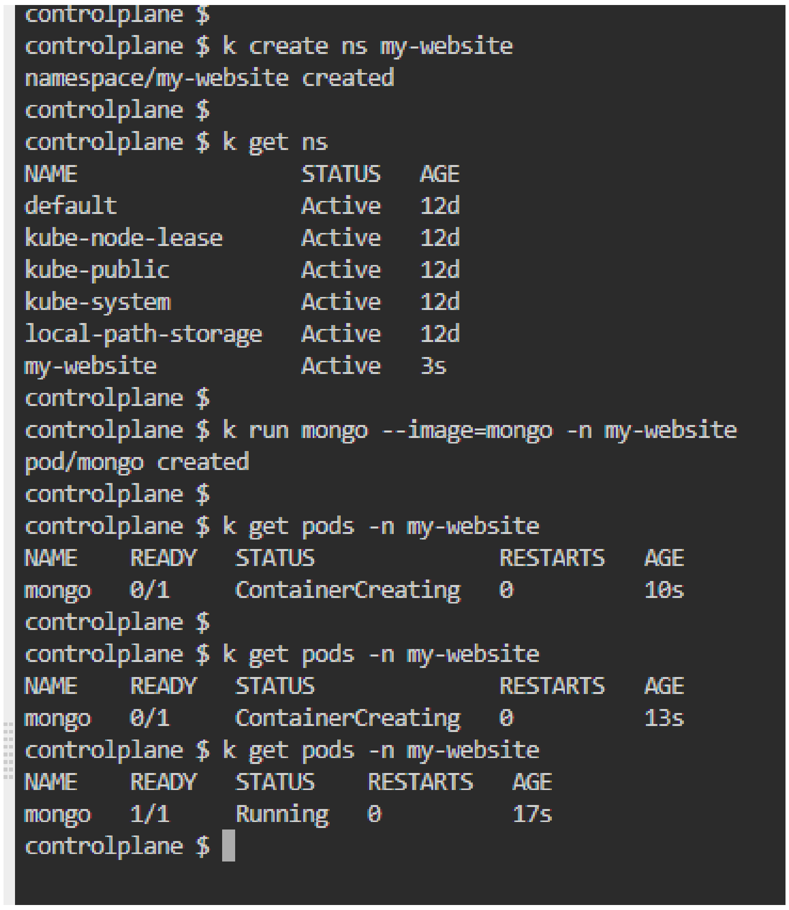
create a deployment named source-ip-app that uses the image registry.k8s.io/echoserver:1.4 .



```
controlplane $
controlplane $ k create deployment source-ip-app --image=registry.k8s.io/echoserver:1.4
deployment.apps/source-ip-app created
controlplane $
controlplane $ k get deployments.apps
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
source-ip-app   0/1     1            0           5s
controlplane $
controlplane $ k get deployments.apps
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
source-ip-app   1/1     1            1           9s
controlplane $
```

Q8

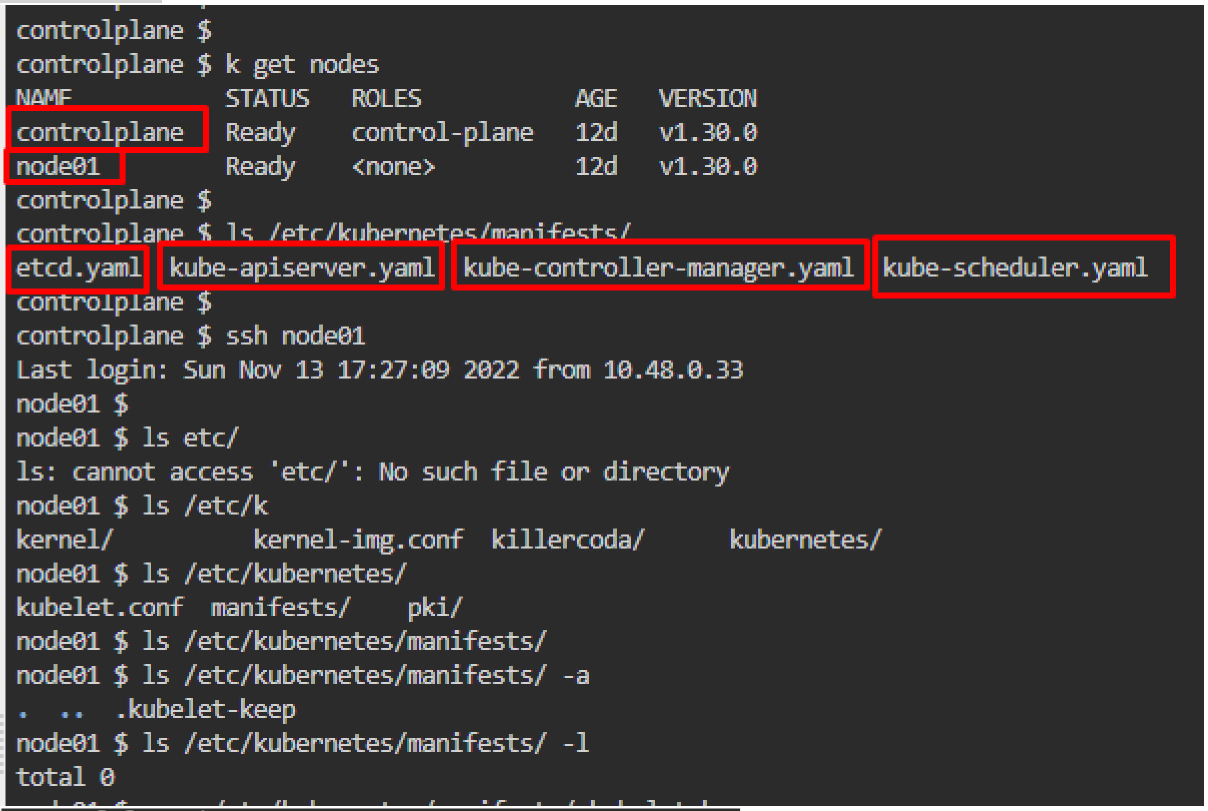
create a pod as follows:
name:mongo
using image:mongo
in anew kubernetes namespacenamed:my-website



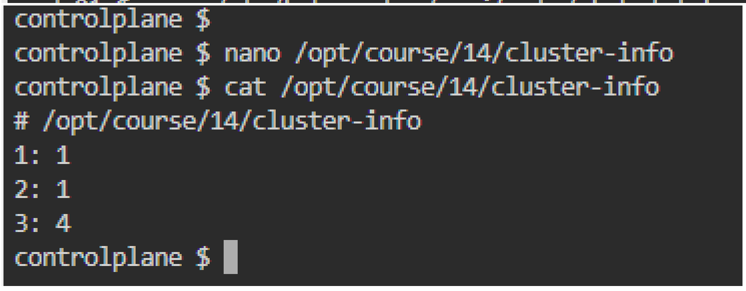
```
controlplane $
controlplane $ k create ns my-website
namespace/my-website created
controlplane $
controlplane $ k get ns
NAME                 STATUS   AGE
default              Active   12d
kube-node-lease      Active   12d
kube-public          Active   12d
kube-system          Active   12d
local-path-storage   Active   12d
my-website           Active   3s
controlplane $
controlplane $ k run mongo --image=mongo -n my-website
pod/mongo created
controlplane $
controlplane $ k get pods -n my-website
NAME    READY   STATUS              RESTARTS   AGE
mongo   0/1     ContainerCreating   0          10s
controlplane $
controlplane $ k get pods -n my-website
NAME    READY   STATUS              RESTARTS   AGE
mongo   0/1     ContainerCreating   0          13s
controlplane $ k get pods -n my-website
NAME    READY   STATUS    RESTARTS   AGE
mongo   1/1     Running   0          17s
controlplane $
```

Q9

You're ask to find out following information about the cluster :
How many controlplane nodes are available?
How many worker nodes are available?
how many static pods are running
Write your answers into file /opt/course/14/cluster-info, structured like this:
/opt/course/14/cluster-info
1: [ANSWER]
2: [ANSWER]
3: [ANSWER]



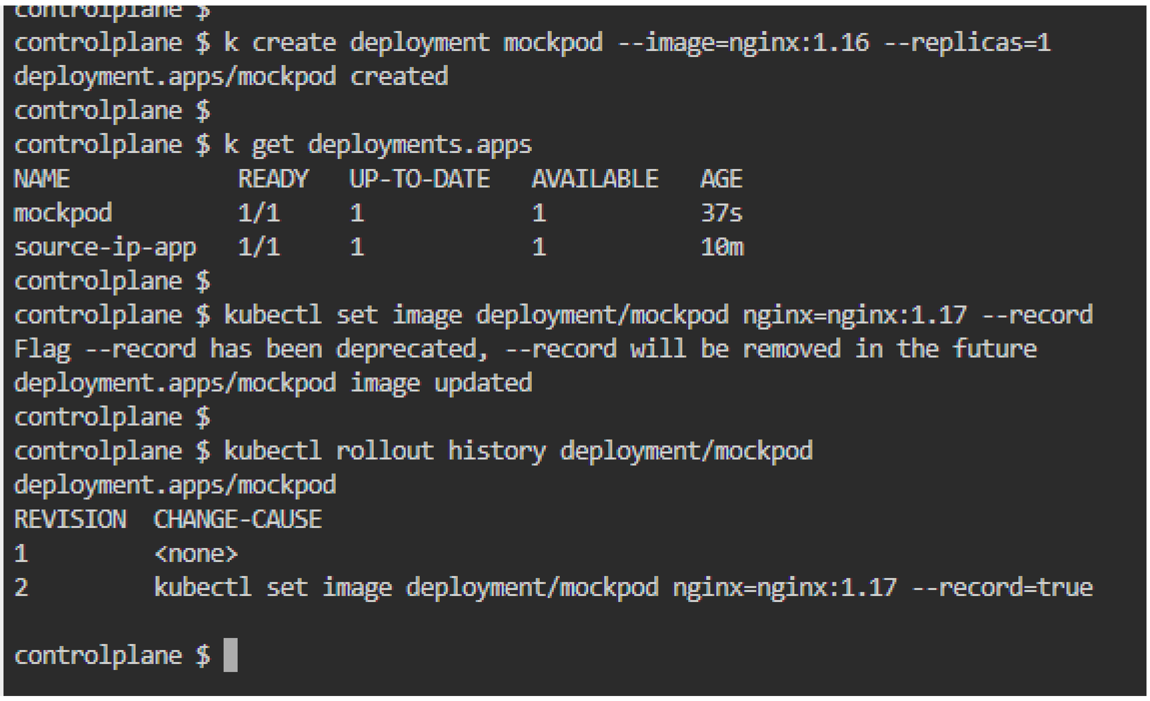
```
controlplane $
controlplane $ k get nodes
NAME           STATUS   ROLES           AGE   VERSION
controlplane   Ready    control-plane   12d   v1.30.0
node01         Ready    <none>          12d   v1.30.0
controlplane $
controlplane $ ls /etc/kubernetes/manifests/
etcd.yaml  kube-apiserver.yaml  kube-controller-manager.yaml  kube-scheduler.yaml
controlplane $
controlplane $ ssh node01
Last login: Sun Nov 13 17:27:09 2022 from 10.48.0.33
node01 $
node01 $ ls etc/
ls: cannot access 'etc/': No such file or directory
node01 $ ls /etc/k
kernel/           kernel-img.conf  killercoda/      kubernetes/
node01 $ ls /etc/kubernetes/
kubelet.conf  manifests/    pki/
node01 $ ls /etc/kubernetes/manifests/
node01 $ ls /etc/kubernetes/manifests/ -a
.  ..  .kubelet-keep
node01 $ ls /etc/kubernetes/manifests/ -l
total 0
```



```
controlplane $
controlplane $ nano /opt/course/14/cluster-info
controlplane $ cat /opt/course/14/cluster-info
# /opt/course/14/cluster-info
1: 1
2: 1
3: 4
controlplane $
```

Q10

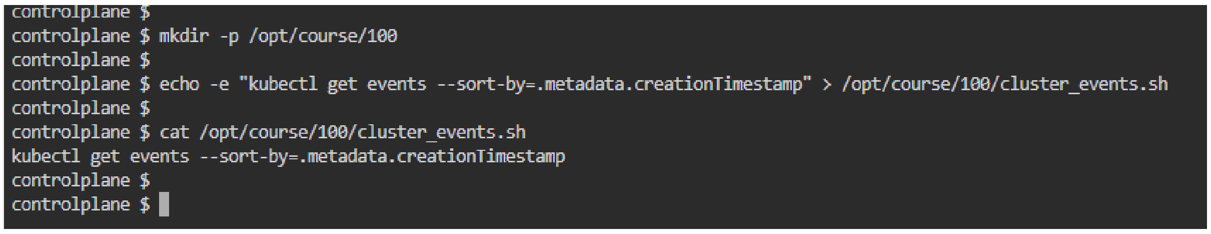
Create a new deployment called mockpod, with image nginx:1.16 and 1 replica.
Next upgrade the deployment to version 1.17 using rolling update
Make sure that the version upgrade is recorded in the resource annotation



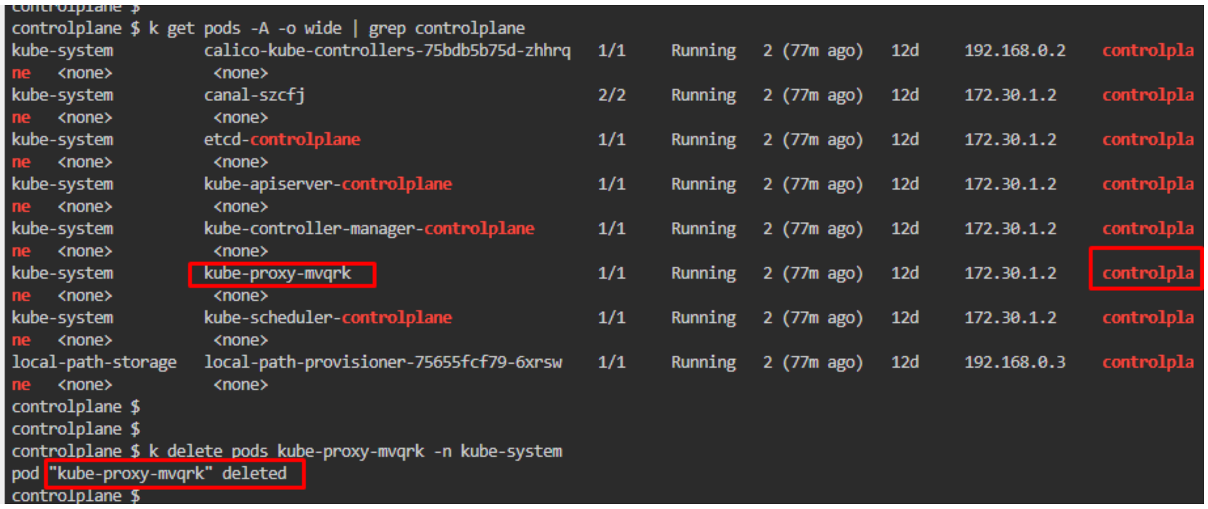
```
controlplane $
controlplane $ k create deployment mockpod --image=nginx:1.16 --replicas=1
deployment.apps/mockpod created
controlplane $
controlplane $ k get deployments.apps
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
mockpod         1/1     1            1           37s
source-ip-app   1/1     1            1           10m
controlplane $
controlplane $ kubectl set image deployment/mockpod nginx=nginx:1.17 --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/mockpod image updated
controlplane $
controlplane $ kubectl rollout history deployment/mockpod
deployment.apps/mockpod
REVISION  CHANGE-CAUSE
1         <none>
2         kubectl set image deployment/mockpod nginx=nginx:1.17 --record=true

controlplane $
```

Q11

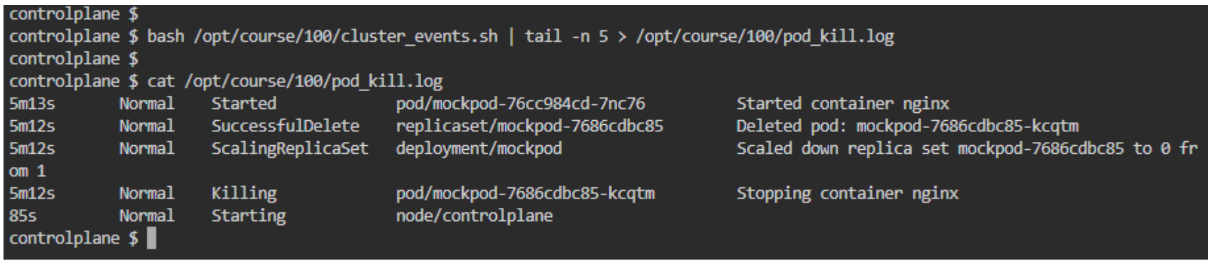
write a command into /opt/course/100/cluster_events.sh which shows the latest events in the whole cluster, ordered by time (metadata.creationtimestamp). use kubectl for it.
now delete the kube-proxy pod running on node controlpane node and write the events this caused
into /opt/course/100/pod_kill.log.



```
controlplane $
controlplane $ mkdir -p /opt/course/100
controlplane $
controlplane $ echo -e "kubectl get events --sort-by=.metadata.creationTimestamp" > /opt/course/100/cluster_events.sh
controlplane $
controlplane $ cat /opt/course/100/cluster_events.sh
kubectl get events --sort-by=.metadata.creationTimestamp
controlplane $
controlplane $
```



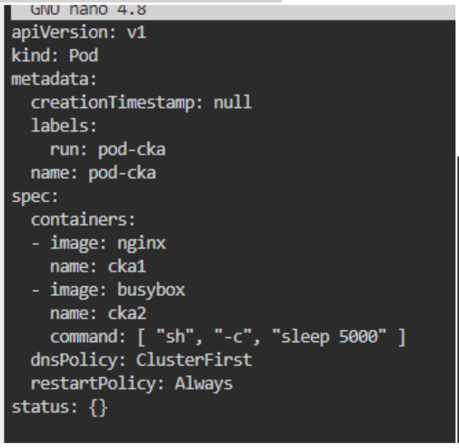
```
controlplane $
controlplane $ k get pods -A -o wide | grep controlplane
kube-system          calico-kube-controllers-75bdb5b75d-zhhrq   1/1     Running   2 (77m ago)   12d     192.168.0.2    controlpla
ne    <none>           <none>
kube-system          canal-szcfj                                2/2     Running   2 (77m ago)   12d     172.30.1.2     controlpla
ne    <none>           <none>
kube-system          etcd-controlplane                          1/1     Running   2 (77m ago)   12d     172.30.1.2     controlpla
ne    <none>           <none>
kube-system          kube-apiserver-controlplane                1/1     Running   2 (77m ago)   12d     172.30.1.2     controlpla
ne    <none>           <none>
kube-system          kube-controller-manager-controlplane       1/1     Running   2 (77m ago)   12d     172.30.1.2     controlpla
ne    <none>           <none>
kube-system          kube-proxy-mvqrk                           1/1     Running   2 (77m ago)   12d     172.30.1.2     controlpla
ne    <none>           <none>
kube-system          kube-scheduler-controlplane                1/1     Running   2 (77m ago)   12d     172.30.1.2     controlpla
ne    <none>           <none>
local-path-storage   local-path-provisioner-75655fcf79-6xrsw    1/1     Running   2 (77m ago)   12d     192.168.0.3    controlpla
ne    <none>           <none>
controlplane $
controlplane $
controlplane $ k delete pods kube-proxy-mvqrk -n kube-system
pod "kube-proxy-mvqrk" deleted
controlplane $
```



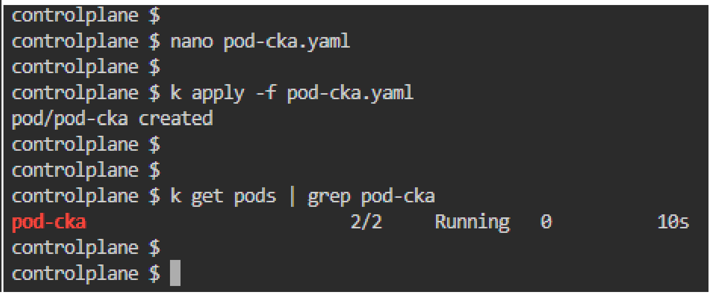
```
controlplane $
controlplane $ bash /opt/course/100/cluster_events.sh | tail -n 5 > /opt/course/100/pod_kill.log
controlplane $
controlplane $ cat /opt/course/100/pod_kill.log
5m13s       Normal    Started             pod/mockpod-76cc984cd-7nc76          Started container nginx
5m12s       Normal    SuccessfulDelete    replicaset/mockpod-7686cdbc85        Deleted pod: mockpod-7686cdbc85-kcqtm
5m12s       Normal    ScalingReplicaSet   deployment/mockpod                   Scaled down replica set mockpod-7686cdbc85 to 0 fr
om 1
5m12s       Normal    Killing             pod/mockpod-7686cdbc85-kcqtm         Stopping container nginx
85s         Normal    Starting            node/controlplane
controlplane $
```

Q12

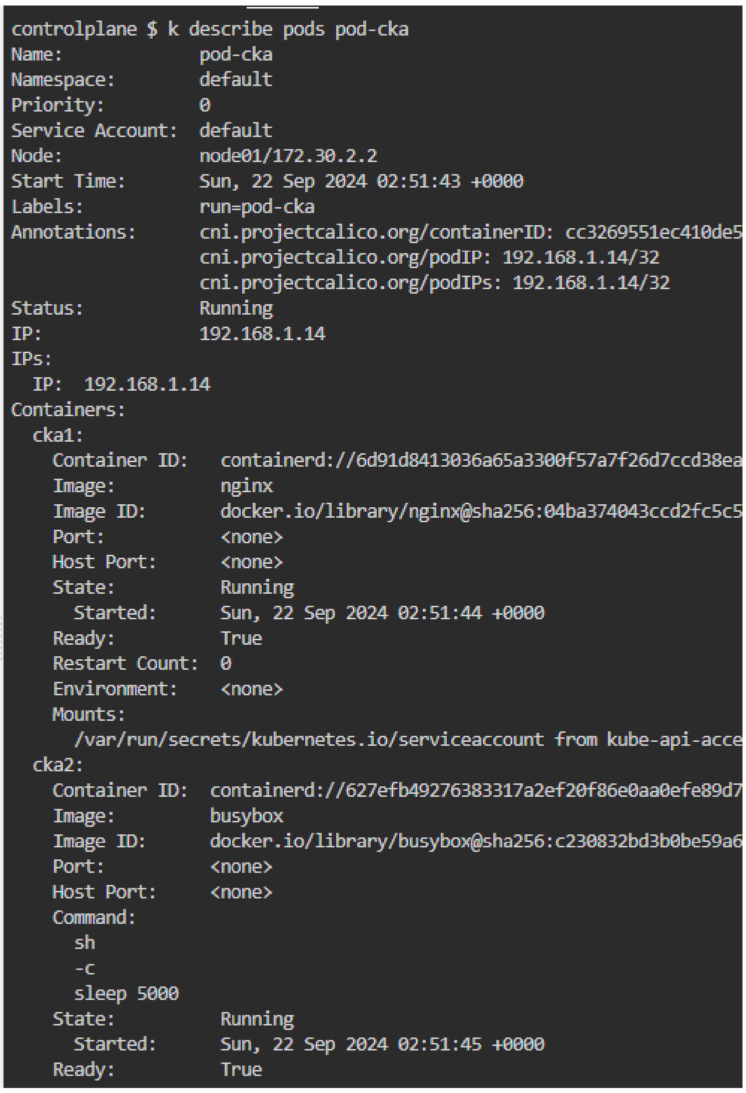
create a deployment called pod-cka with two containers, as given below:
container 1 - name: cka1, image: nginx
container2 - name: cka2, image:busybox,
command: sleep 5000



```
  GNU nano 4.8
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: pod-cka
  name: pod-cka
spec:
  containers:
  - image: nginx
    name: cka1
  - image: busybox
    name: cka2
    command: [ "sh", "-c", "sleep 5000" ]
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```



```
controlplane $
controlplane $ nano pod-cka.yaml
controlplane $
controlplane $ k apply -f pod-cka.yaml
pod/pod-cka created
controlplane $
controlplane $
controlplane $ k get pods | grep pod-cka
pod-cka                          2/2     Running   0          10s
controlplane $
controlplane $
```

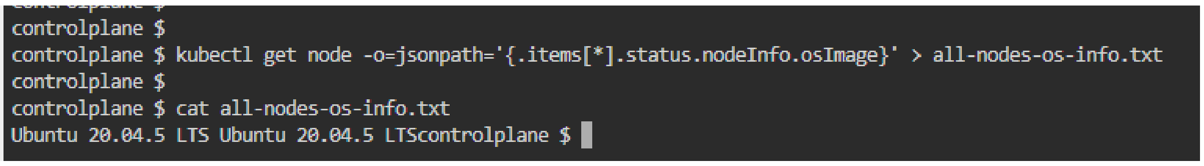


```
controlplane $ k describe pods pod-cka
Name:             pod-cka
Namespace:        default
Priority:         0
Service Account:  default
Node:             node01/172.30.2.2
Start Time:       Sun, 22 Sep 2024 02:51:43 +0000
Labels:           run=pod-cka
Annotations:      cni.projectcalico.org/containerID: cc3269551ec410de5
                  cni.projectcalico.org/podIP: 192.168.1.14/32
                  cni.projectcalico.org/podIPs: 192.168.1.14/32
Status:           Running
IP:               192.168.1.14
IPs:
  IP:  192.168.1.14
Containers:
  cka1:
    Container ID:   containerd://6d91d8413036a65a3300f57a7f26d7ccd38ea
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:04ba374043ccd2fc5c5
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sun, 22 Sep 2024 02:51:44 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-acce
  cka2:
    Container ID:  containerd://627efb49276383317a2ef20f86e0aa0efe89d7
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:c230832bd3b0be59a6
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
      sleep 5000
    State:          Running
      Started:      Sun, 22 Sep 2024 02:51:45 +0000
    Ready:          True
```

Q13

use json path query to retrieve the osimages of all the nodes and store it in a file “all-nodes-os-info.txt” at root location.

note: the osimage are under the nodeInfo section under status of each node.



```
controlplane $
controlplane $ kubectl get node -o=jsonpath='{.items[*].status.nodeInfo.osImage}' > all-nodes-os-info.txt
controlplane $
controlplane $ cat all-nodes-os-info.txt
Ubuntu 20.04.5 LTS Ubuntu 20.04.5 LTScontrolplane $
```

Q14

create a new persistentvolumeclaim:

☞ name: pv-volume

☞ class: csi-hostpath-sc

☞ capacity: 10mi

create a new pod which mounts the persistentvolumeclaim as a volume:

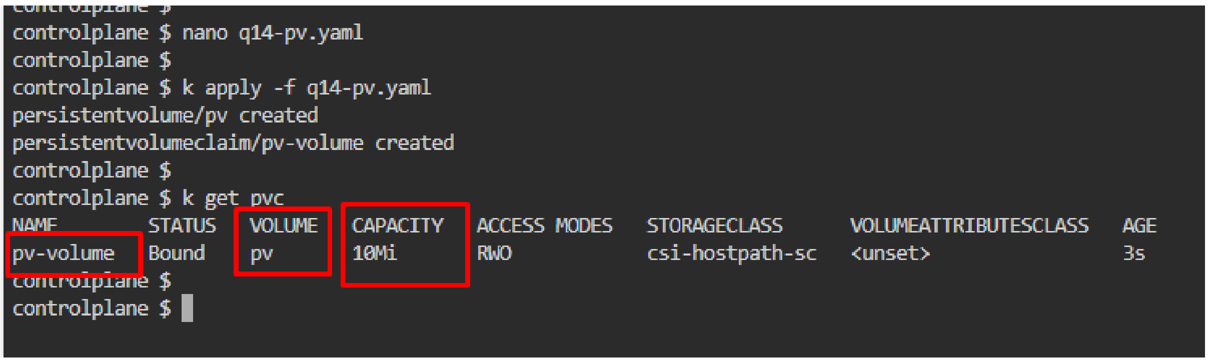
☞ name: web-server

☞ image: nginx

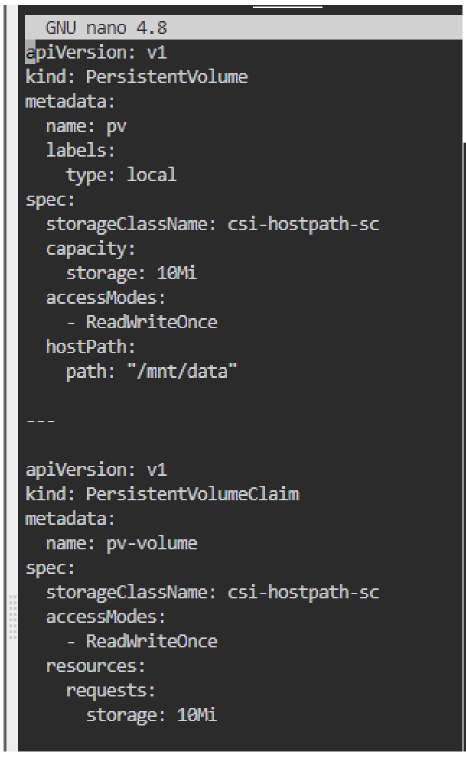
☞ mount path: /usr/share/nginx/html

configure the new pod to have readwriteonce access on the volume.

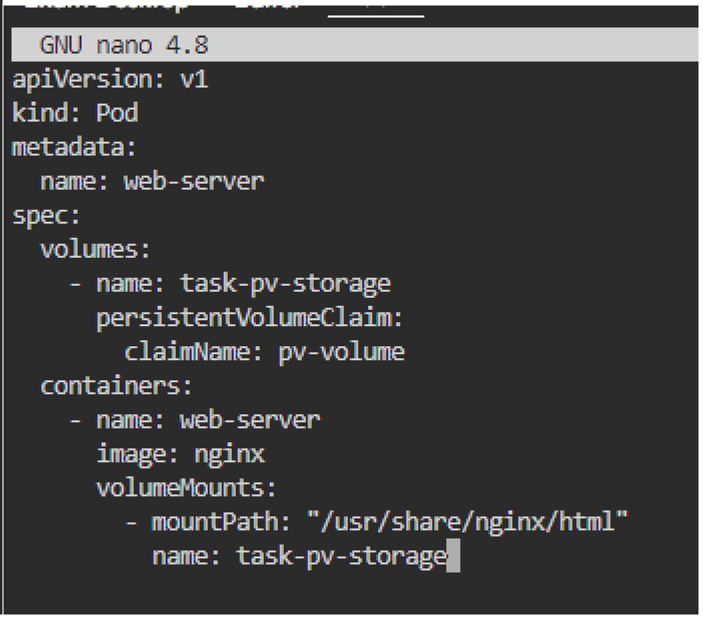
finally, using kubectl edit or kubectl patch expand the persistentvolumeclaim to a capacity of 70mi and record that change.



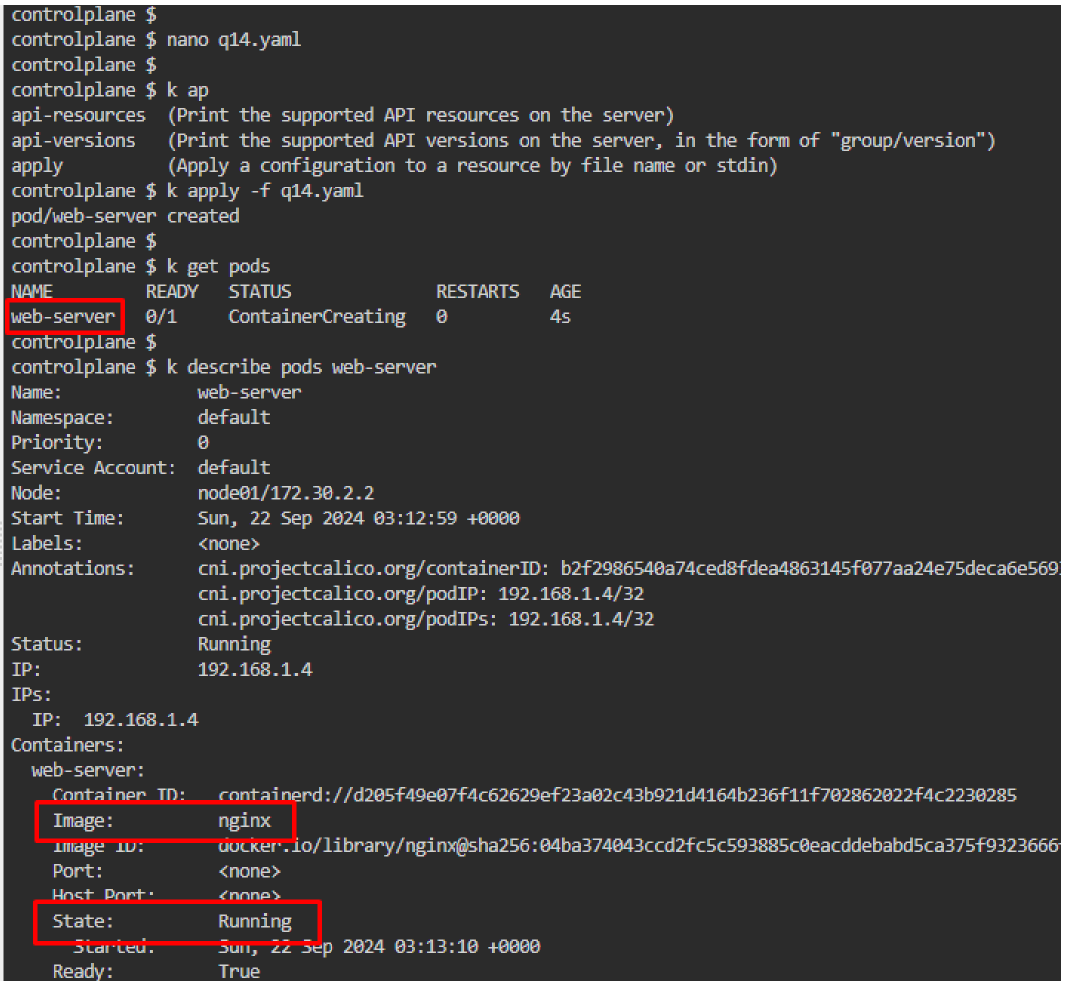
```
controlplane $ nano q14-pv.yaml
controlplane $
controlplane $ k apply -f q14-pv.yaml
persistentvolume/pv created
persistentvolumeclaim/pv-volume created
controlplane $
controlplane $ k get pvc
NAME        STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS      VOLUMEATTRIBUTESCLASS   AGE
pv-volume   Bound    pv       10Mi       RWO            csi-hostpath-sc   <unset>                 3s
controlplane $
controlplane $
```



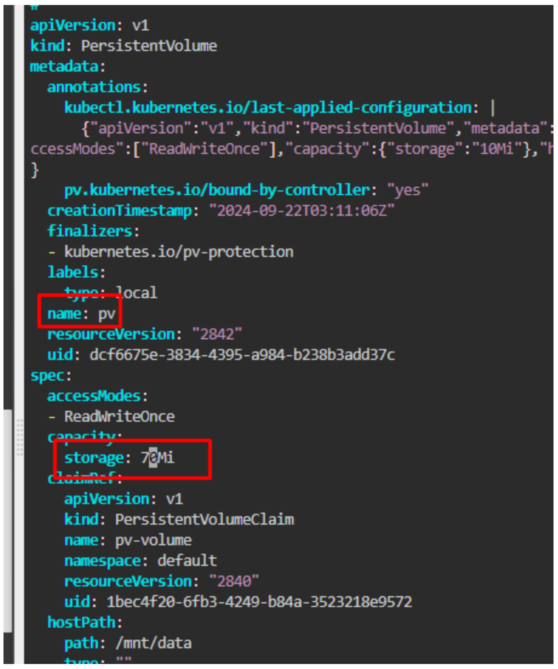
```
  GNU nano 4.8
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv
  labels:
    type: local
spec:
  storageClassName: csi-hostpath-sc
  capacity:
    storage: 10Mi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"

---

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-volume
spec:
  storageClassName: csi-hostpath-sc
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Mi
```



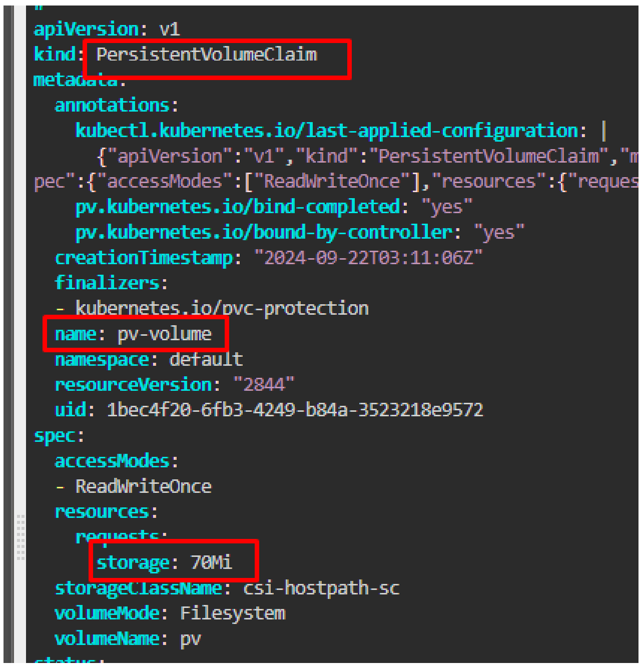
```
  GNU nano 4.8
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: pv-volume
  containers:
    - name: web-server
      image: nginx
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
```



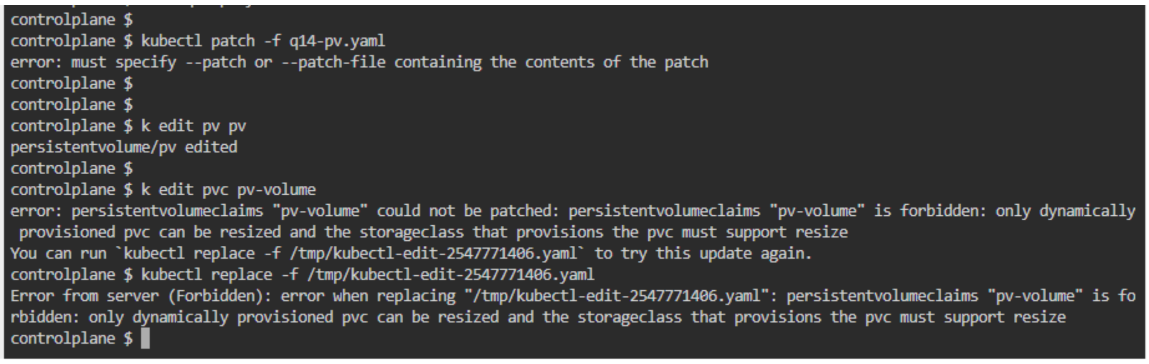
```
controlplane $
controlplane $ nano q14.yaml
controlplane $
controlplane $ k ap
api-resources  (Print the supported API resources on the server)
api-versions   (Print the supported API versions on the server, in the form of "group/version")
apply          (Apply a configuration to a resource by file name or stdin)
controlplane $ k apply -f q14.yaml
pod/web-server created
controlplane $
controlplane $ k get pods
NAME         READY   STATUS              RESTARTS   AGE
web-server   0/1     ContainerCreating   0          4s
controlplane $
controlplane $ k describe pods web-server
Name:             web-server
Namespace:        default
Priority:         0
Service Account:  default
Node:             node01/172.30.2.2
Start Time:       Sun, 22 Sep 2024 03:12:59 +0000
Labels:           <none>
Annotations:      cni.projectcalico.org/containerID: b2f2986540a74ced8fdea4863145f077aa24e75deca6e569
                  cni.projectcalico.org/podIP: 192.168.1.4/32
                  cni.projectcalico.org/podIPs: 192.168.1.4/32
Status:           Running
IP:               192.168.1.4
IPs:
  IP:  192.168.1.4
Containers:
  web-server:
    Container ID:   containerd://d205f49e07f4c62629ef23a02c43b921d4164b236f11f702862022f4c2230285
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:04ba374043ccd2fc5c593885c0eacddebabd5ca375f9323666
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sun, 22 Sep 2024 03:13:10 +0000
    Ready:          True
```



```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"PersistentVolume","metadata":
ccessModes":["ReadWriteOnce"],"capacity":{"storage":"10Mi"},"h
}
    pv.kubernetes.io/bound-by-controller: "yes"
  creationTimestamp: "2024-09-22T03:11:06Z"
  finalizers:
  - kubernetes.io/pv-protection
  labels:
    type: local
  name: pv
  resourceVersion: "2842"
  uid: dcf6675e-3834-4395-a984-b238b3add37c
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 70Mi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: pv-volume
    namespace: default
    resourceVersion: "2840"
    uid: 1bec4f20-6fb3-4249-b84a-3523218e9572
  hostPath:
    path: /mnt/data
    type: ""
```



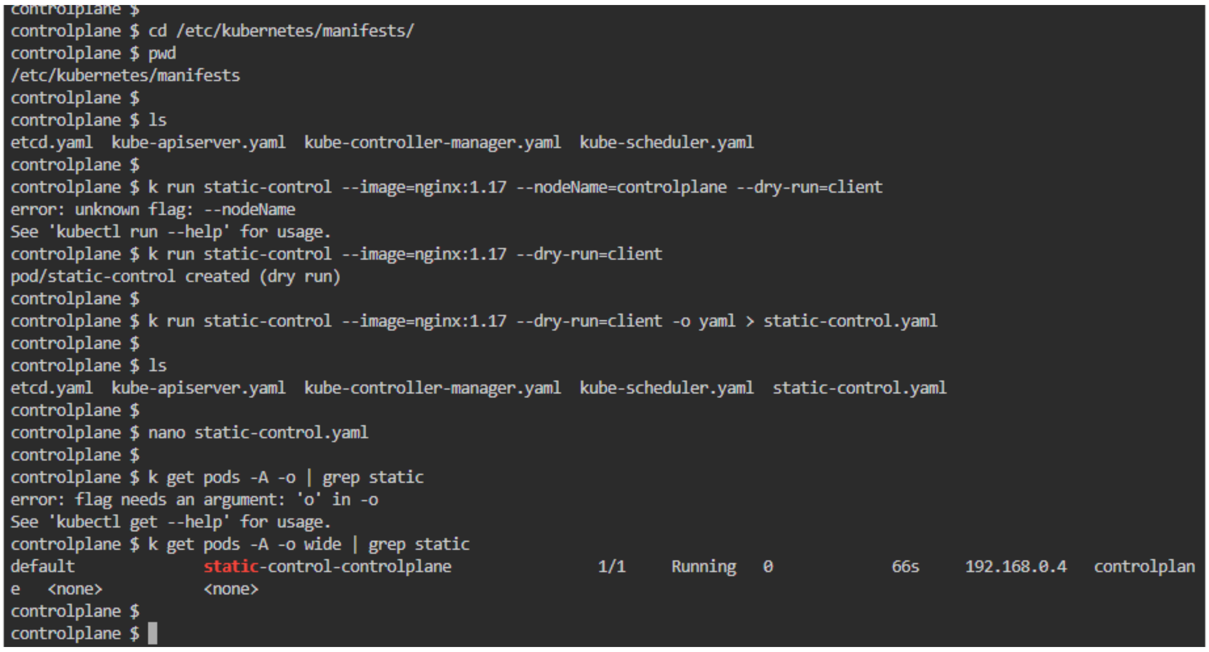
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"PersistentVolumeClaim","m
pec":{"accessModes":["ReadWriteOnce"],"resources":{"reques
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
  creationTimestamp: "2024-09-22T03:11:06Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: pv-volume
  namespace: default
  resourceVersion: "2844"
  uid: 1bec4f20-6fb3-4249-b84a-3523218e9572
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 70Mi
  storageClassName: csi-hostpath-sc
  volumeMode: Filesystem
  volumeName: pv
status:
```



```
controlplane $
controlplane $ kubectl patch -f q14-pv.yaml
error: must specify --patch or --patch-file containing the contents of the patch
controlplane $
controlplane $
controlplane $ k edit pv pv
persistentvolume/pv edited
controlplane $
controlplane $ k edit pvc pv-volume
error: persistentvolumeclaims "pv-volume" could not be patched: persistentvolumeclaims "pv-volume" is forbidden: only dynamically
 provisioned pvc can be resized and the storageclass that provisions the pvc must support resize
You can run `kubectl replace -f /tmp/kubectl-edit-2547771406.yaml` to try this update again.
controlplane $ kubectl replace -f /tmp/kubectl-edit-2547771406.yaml
Error from server (Forbidden): error when replacing "/tmp/kubectl-edit-2547771406.yaml": persistentvolumeclaims "pv-volume" is fo
rbidden: only dynamically provisioned pvc can be resized and the storageclass that provisions the pvc must support resize
controlplane $ 
```

Q15

create a static pod named static-control on the control plane node that uses the nginx:1.17



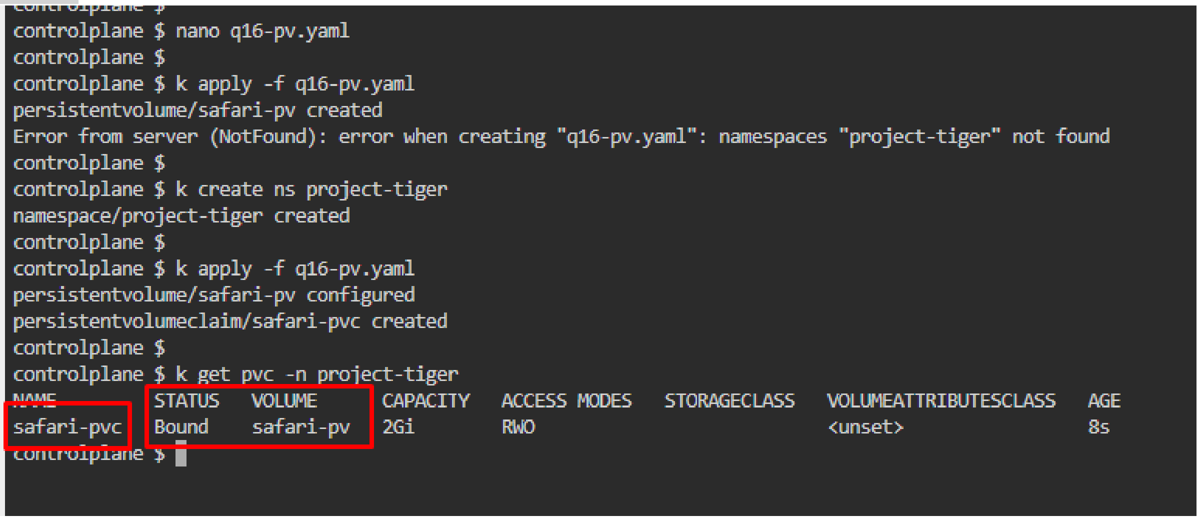
```
controlplane $
controlplane $ cd /etc/kubernetes/manifests/
controlplane $ pwd
/etc/kubernetes/manifests
controlplane $
controlplane $ ls
etcd.yaml  kube-apiserver.yaml  kube-controller-manager.yaml  kube-scheduler.yaml
controlplane $
controlplane $ k run static-control --image=nginx:1.17 --nodeName=controlplane --dry-run=client
error: unknown flag: --nodeName
See 'kubectl run --help' for usage.
controlplane $ k run static-control --image=nginx:1.17 --dry-run=client
pod/static-control created (dry run)
controlplane $
controlplane $ k run static-control --image=nginx:1.17 --dry-run=client -o yaml > static-control.yaml
controlplane $
controlplane $ ls
etcd.yaml  kube-apiserver.yaml  kube-controller-manager.yaml  kube-scheduler.yaml  static-control.yaml
controlplane $
controlplane $ nano static-control.yaml
controlplane $
controlplane $ k get pods -A -o | grep static
error: flag needs an argument: 'o' in -o
See 'kubectl get --help' for usage.
controlplane $ k get pods -A -o wide | grep static
default            static-control-controlplane             1/1     Running   0          66s     192.168.0.4    controlplan
e   <none>         <none>
controlplane $
controlplane $ 
```

Q16

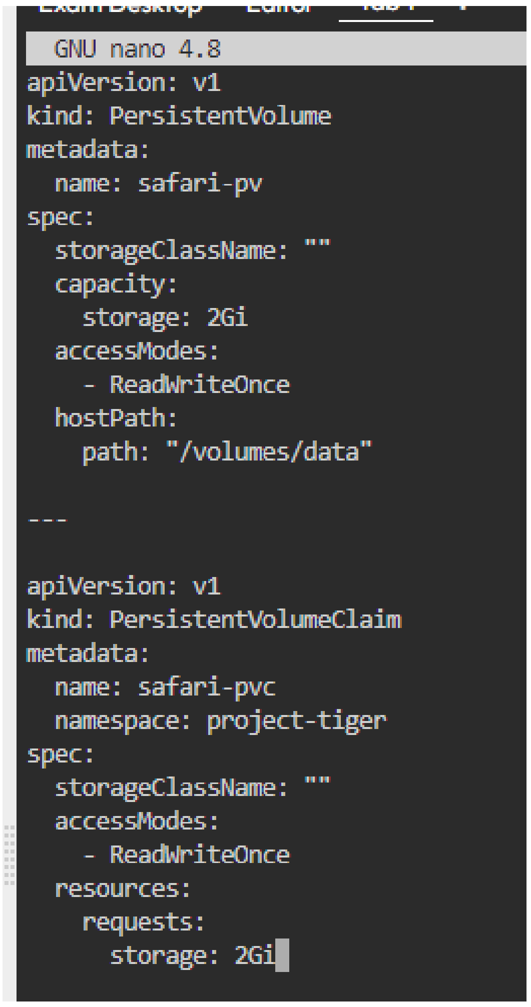
Create a new PersistentVolume named safari-pv. It should have a capacity of 2Gi, accessMode ReadWriteOnce, hostPath /Volumes/Data and no storageClassName defined.

Next create a new PersistentVolumeClaim in Namespace project-tiger named safari-pvc . It should request 2Gi storage, accessMode ReadWriteOnce and should not define a storageClassName. The PVC should bound to the PV correctly.

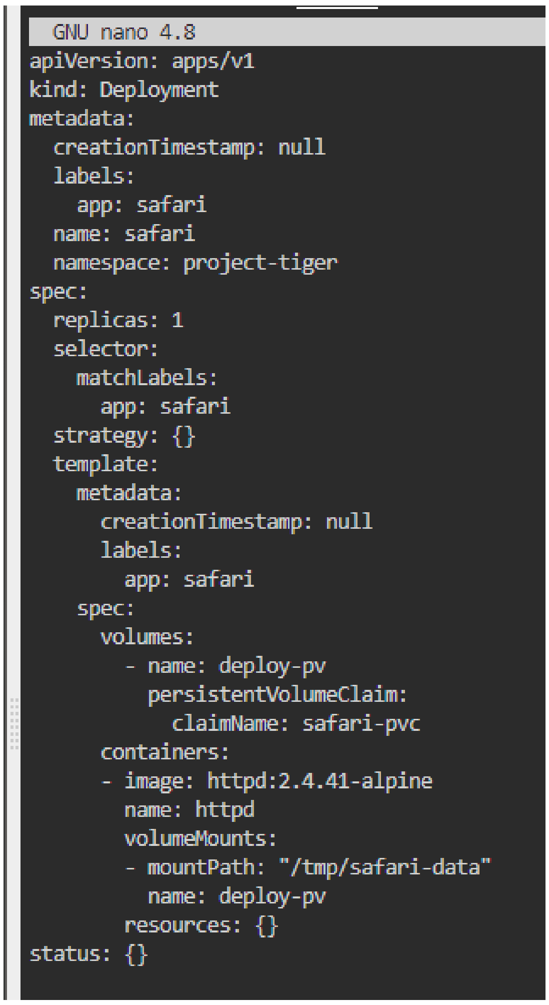
Finally create a new Deployment safari in Namespace project-tiger which mounts that volume at /tmp/safari-data. The Pods of that Deployment should be of image httpd:2.4.41-alpine.



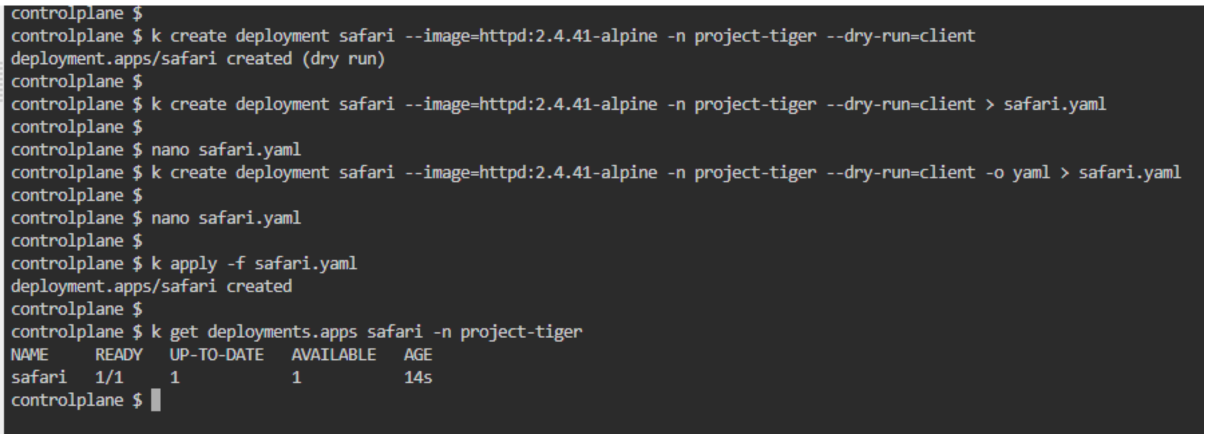
```
controlplane $
controlplane $ nano q16-pv.yaml
controlplane $
controlplane $ k apply -f q16-pv.yaml
persistentvolume/safari-pv created
Error from server (NotFound): error when creating "q16-pv.yaml": namespaces "project-tiger" not found
controlplane $
controlplane $ k create ns project-tiger
namespace/project-tiger created
controlplane $
controlplane $ k apply -f q16-pv.yaml
persistentvolume/safari-pv configured
persistentvolumeclaim/safari-pvc created
controlplane $
controlplane $ k get pvc -n project-tiger
NAME         STATUS   VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
safari-pvc   Bound    safari-pv   2Gi        RWO                           <unset>                 8s
controlplane $ 
```



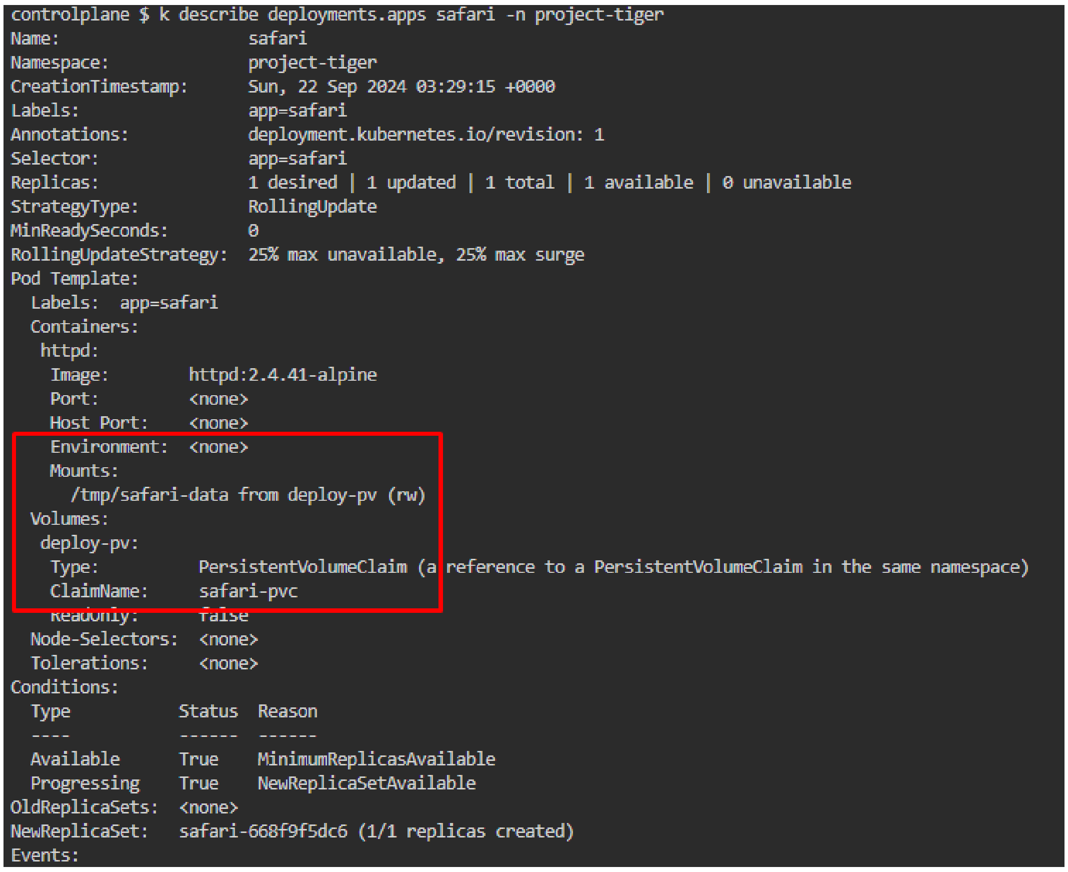
```
  GNU nano 4.8
apiVersion: v1
kind: PersistentVolume
metadata:
  name: safari-pv
spec:
  storageClassName: ""
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/volumes/data"

---

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: safari-pvc
  namespace: project-tiger
spec:
  storageClassName: ""
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```



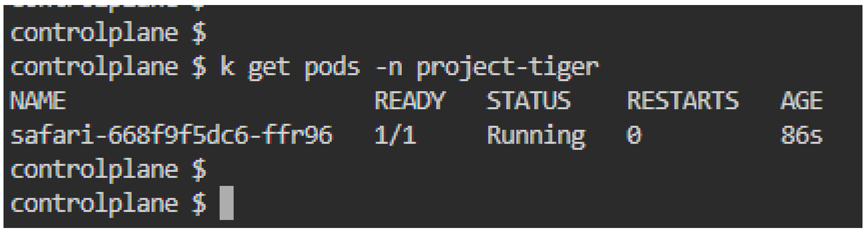
```
  GNU nano 4.8
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: safari
  name: safari
  namespace: project-tiger
spec:
  replicas: 1
  selector:
    matchLabels:
      app: safari
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: safari
    spec:
      volumes:
        - name: deploy-pv
          persistentVolumeClaim:
            claimName: safari-pvc
      containers:
      - image: httpd:2.4.41-alpine
        name: httpd
        volumeMounts:
        - mountPath: "/tmp/safari-data"
          name: deploy-pv
        resources: {}
status: {}
```



```
controlplane $
controlplane $ k create deployment safari --image=httpd:2.4.41-alpine -n project-tiger --dry-run=client
deployment.apps/safari created (dry run)
controlplane $
controlplane $ k create deployment safari --image=httpd:2.4.41-alpine -n project-tiger --dry-run=client > safari.yaml
controlplane $
controlplane $ nano safari.yaml
controlplane $ k create deployment safari --image=httpd:2.4.41-alpine -n project-tiger --dry-run=client -o yaml > safari.yaml
controlplane $
controlplane $ nano safari.yaml
controlplane $
controlplane $ k apply -f safari.yaml
deployment.apps/safari created
controlplane $
controlplane $ k get deployments.apps safari -n project-tiger
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
safari   1/1     1            1           14s
controlplane $
```



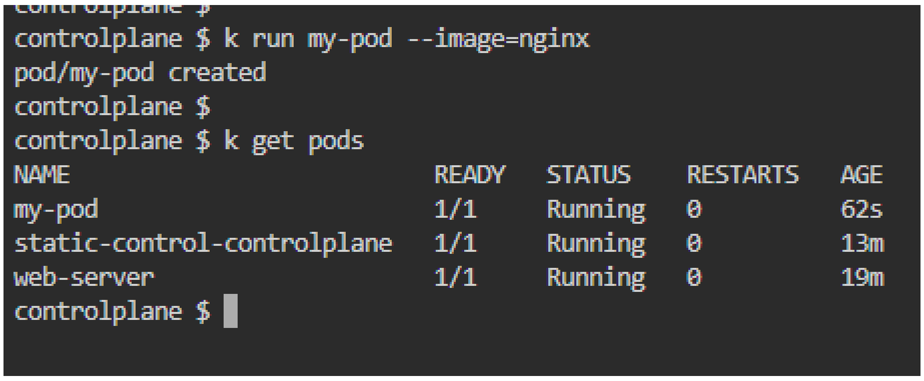
```
controlplane $ k describe deployments.apps safari -n project-tiger
Name:                   safari
Namespace:              project-tiger
CreationTimestamp:      Sun, 22 Sep 2024 03:29:15 +0000
Labels:                 app=safari
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=safari
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:  app=safari
  Containers:
   httpd:
    Image:        httpd:2.4.41-alpine
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:
      /tmp/safari-data from deploy-pv (rw)
  Volumes:
   deploy-pv:
    Type:          PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:     safari-pvc
    ReadOnly:      false
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   safari-668f9f5dc6 (1/1 replicas created)
Events:
```



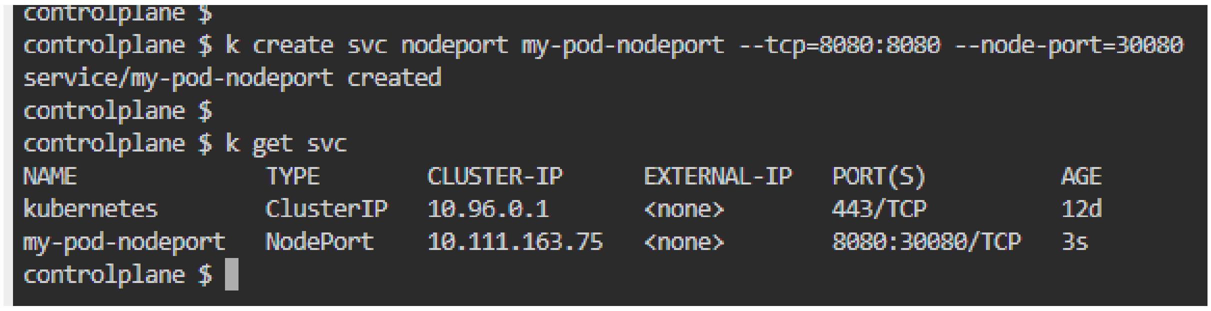
```
controlplane $
controlplane $ k get pods -n project-tiger
NAME                      READY   STATUS    RESTARTS   AGE
safari-668f9f5dc6-ffr96   1/1     Running   0          86s
controlplane $
controlplane $
```

Q17

Create a NodePort service to expose a pod named **my-pod** on port 8080, with the NodePort set to
30080.



```
controlplane $ k run my-pod --image=nginx
pod/my-pod created
controlplane $
controlplane $ k get pods
NAME                          READY   STATUS    RESTARTS   AGE
my-pod                        1/1     Running   0          62s
static-control-controlplane   1/1     Running   0          13m
web-server                    1/1     Running   0          19m
controlplane $
```



```
controlplane $
controlplane $ k create svc nodeport my-pod-nodeport --tcp=8080:8080 --node-port=30080
service/my-pod-nodeport created
controlplane $
controlplane $ k get svc
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP          12d
my-pod-nodeport   NodePort    10.111.163.75   <none>        8080:30080/TCP   3s
controlplane $
```