CKA Exam Questions and Answers

Q1) Create a new service account with the name pvviewer. Grant this Service account access to list all PersistentVolumes in the cluster by creating an appropriate cluster role called pvviewer-role and ClusterRoleBinding called pvviewer-role-binding.

Next, create a pod called **pvviewer** with the **image: redis** and **serviceaccount: pvviewer** in the default namespace.

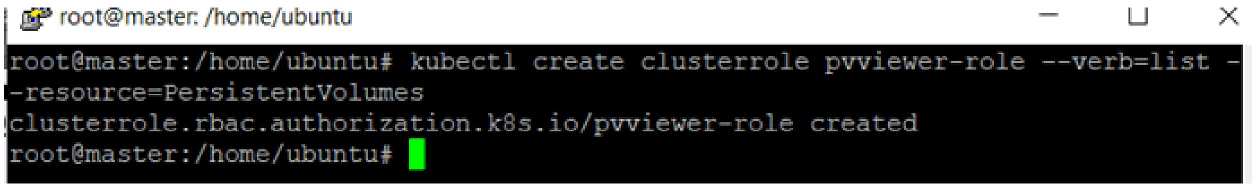
Ans.

Create Service account

```
$ kubectl create serviceaccount pvviewer
```

Create cluster role

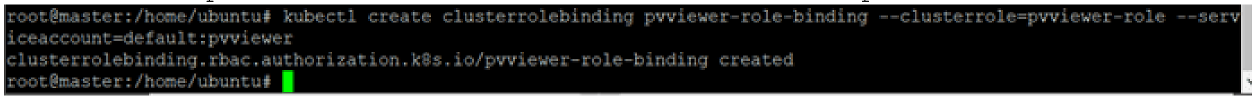
```
$ kubectl create clusterrole pvviewer-role --verb=list --
resource=PersistentVolumes
```



```
root@master:/home/ubuntu# kubectl create clusterrole pvviewer-role --verb=list -
-resource=PersistentVolumes
clusterrole.rbac.authorization.k8s.io/pvviewer-role created
root@master:/home/ubuntu#
```

Create cluster role binding

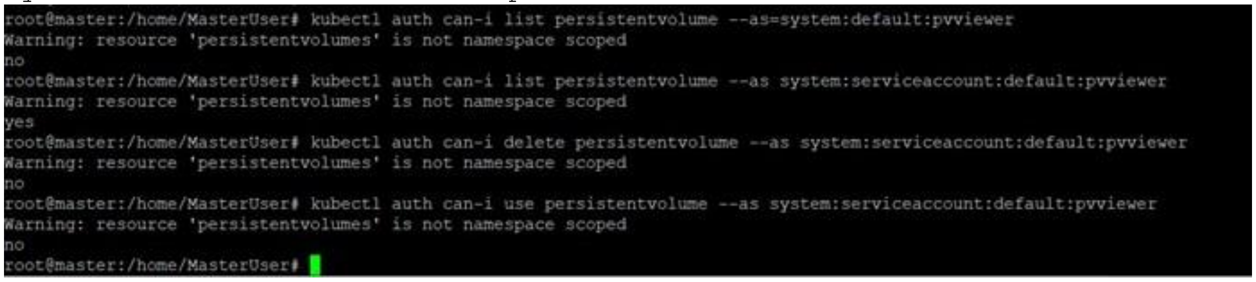
```
$ kubectl create clusterrolebinding pvviewer-role-binding --
clusterrole=pvviewer-role --serviceaccount=default:pvviewer
```



```
root@master:/home/ubuntu# kubectl create clusterrolebinding pvviewer-role-binding --clusterrole=pvviewer-role --serv
iceaccount=default:pvviewer
clusterrolebinding.rbac.authorization.k8s.io/pvviewer-role-binding created
root@master:/home/ubuntu#
```

Verify

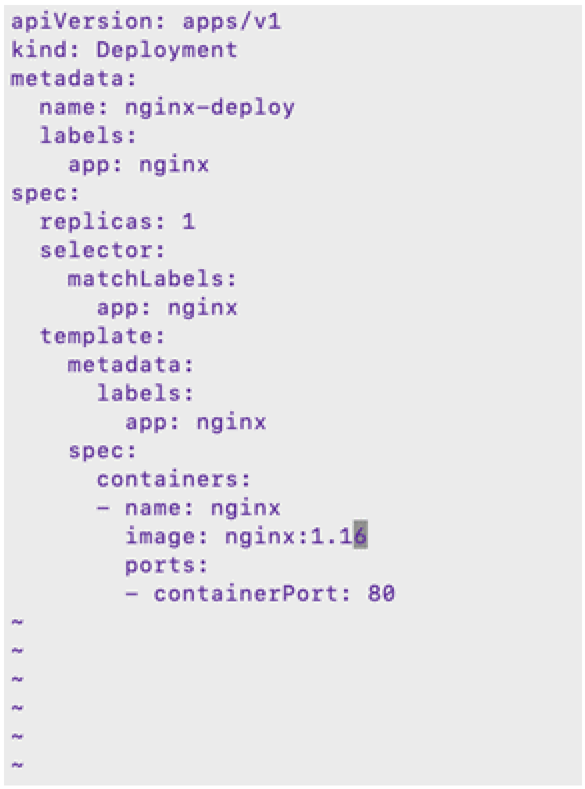
```
$ kubectl auth can-i list PersistentVolumes –as
system:serviceaccount:default:pvviewer
```



```
root@master:/home/MasterUser# kubectl auth can-i list persistentvolume --as=system:default:pvviewer
Warning: resource 'persistentvolumes' is not namespace scoped
no
root@master:/home/MasterUser# kubectl auth can-i list persistentvolume --as system:serviceaccount:default:pvviewer
Warning: resource 'persistentvolumes' is not namespace scoped
yes
root@master:/home/MasterUser# kubectl auth can-i delete persistentvolume --as system:serviceaccount:default:pvviewer
Warning: resource 'persistentvolumes' is not namespace scoped
no
root@master:/home/MasterUser# kubectl auth can-i use persistentvolume --as system:serviceaccount:default:pvviewer
Warning: resource 'persistentvolumes' is not namespace scoped
no
root@master:/home/MasterUser#
```

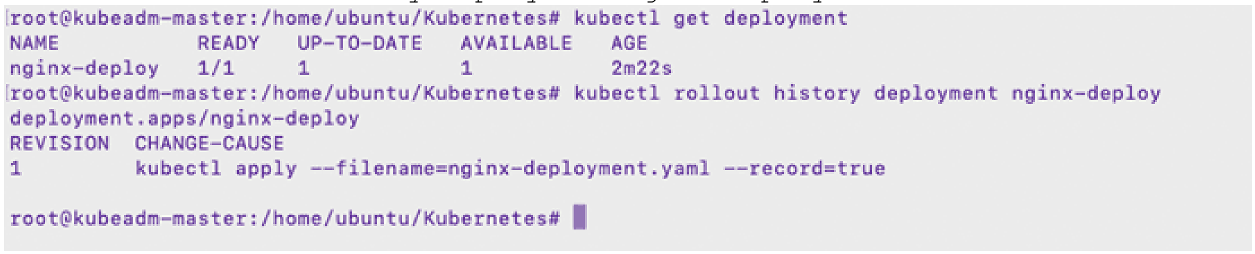
Q2) Create a new deployment called nginx-deploy, with image nginx:1.16 and 1 replica. Record the version. Next upgrade the deployment to version 1.17 using rolling update. Make sure that the version upgrade is recorded in the resource annotation.

Ans.



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deploy
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16
        ports:
        - containerPort: 80
~
~
~
~
~
~
```

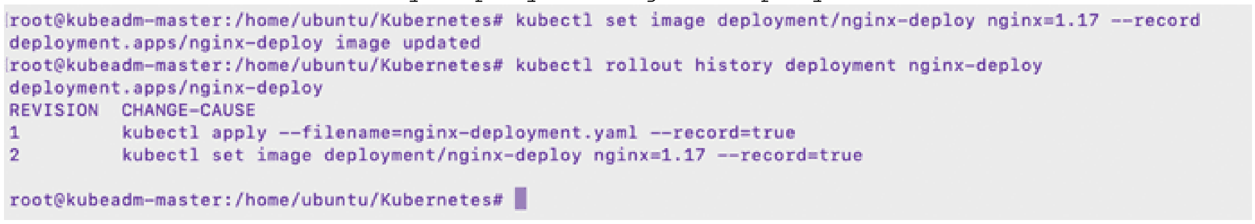
```
$ vim nginx-deployment.yaml
$ kubectl apply -f nginx-deployment.yaml --record
$ kubectl get deployment
$ kubectl rollout history deployment nginx-deploy
```



```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deploy   1/1     1            1           2m22s
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl rollout history deployment nginx-deploy
deployment.apps/nginx-deploy
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=nginx-deployment.yaml --record=true

root@kubeadm-master:/home/ubuntu/Kubernetes#
```

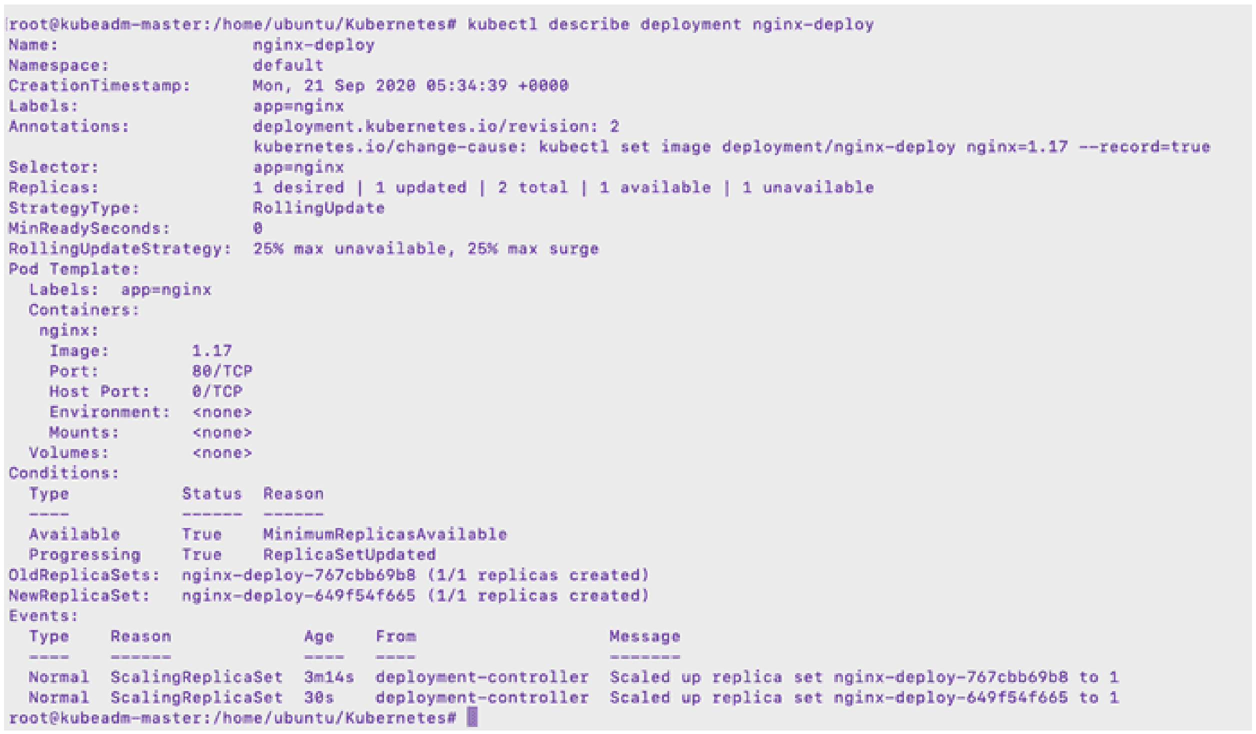
```
$ kubectl set image deployment/nginx-deploy nginx=1.17 --record
$ kubectl rollout history deployment nginx-deploy
```



```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl set image deployment/nginx-deploy nginx=1.17 --record
deployment.apps/nginx-deploy image updated
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl rollout history deployment nginx-deploy
deployment.apps/nginx-deploy
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=nginx-deployment.yaml --record=true
2         kubectl set image deployment/nginx-deploy nginx=1.17 --record=true

root@kubeadm-master:/home/ubuntu/Kubernetes#
```

```
$ kubectl describe deployment nginx-deploy
```



```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe deployment nginx-deploy
Name:                   nginx-deploy
Namespace:              default
CreationTimestamp:      Mon, 21 Sep 2020 05:34:39 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision: 2
                        kubernetes.io/change-cause: kubectl set image deployment/nginx-deploy nginx=1.17 --record=true
Selector:               app=nginx
Replicas:               1 desired | 1 updated | 2 total | 1 available | 1 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        1.17
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    ReplicaSetUpdated
OldReplicaSets:  nginx-deploy-767cbb69b8 (1/1 replicas created)
NewReplicaSet:   nginx-deploy-649f54f665 (1/1 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  3m14s  deployment-controller  Scaled up replica set nginx-deploy-767cbb69b8 to 1
  Normal  ScalingReplicaSet  30s    deployment-controller  Scaled up replica set nginx-deploy-649f54f665 to 1
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

Q3) Create snapshot of the etcd running at https://127.0.0.1:2379. Save snapshot into /opt/etcd-snapshot.db.

Use these are certificate for snapshot

```
Ca certificate: /etc/kubernetes/pki/etcd/ca.crt
Client certicate: /etc/kubernetes/pki/etcd/server.crt
client key: /etc/kubernetes/pki/etcd/server.key
```

and then restore from the previous ETCD backup.

Ans:

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

Verify//

Note: Do not perform this step in exam otherwise it may create an issue in the restoration process.

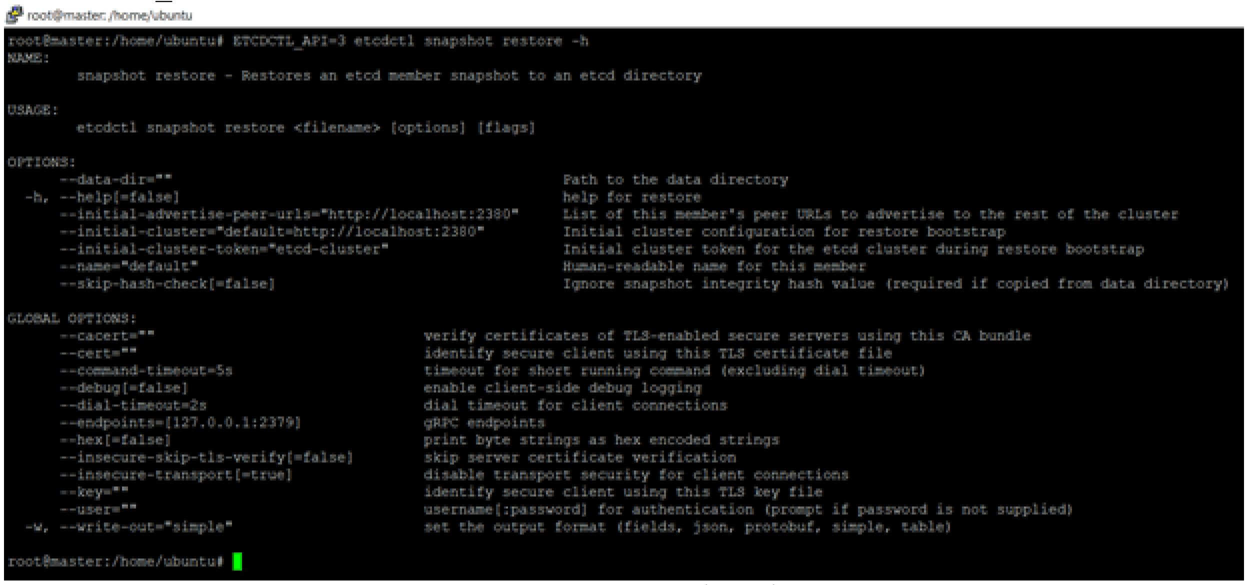
```
$ ETCDCTL_API=3 etcdctl --write-out=table snapshot status /opt/etcd-
snapshot.db
```

Restore

No need to remember all the flags in the restore command:

You can do

```
$ ETCDCTL_API=3 etcdctl snapshot restore -h
```



```
root@master: /home/ubuntu
root@master:/home/ubuntu# ETCDCTL_API=3 etcdctl snapshot restore -h
NAME:
        snapshot restore - Restores an etcd member snapshot to an etcd directory

USAGE:
        etcdctl snapshot restore <filename> [options] [flags]

OPTIONS:
      --data-dir=""                                       Path to the data directory
  -h, --help[=false]                                      help for restore
      --initial-advertise-peer-urls="http://localhost:2380"     List of this member's peer URLs to advertise to the rest of the cluster
      --initial-cluster="default=http://localhost:2380"   Initial cluster configuration for restore bootstrap
      --initial-cluster-token="etcd-cluster"              Initial cluster token for the etcd cluster during restore bootstrap
      --name="default"                                    Human-readable name for this member
      --skip-hash-check[=false]                           Ignore snapshot integrity hash value (required if copied from data directory)

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 timeout)
      --debug[=false]                         enable client-side debug logging
      --dial-timeout=2s                       dial timeout for client connections
      --endpoints=[127.0.0.1:2379]            gRPC endpoints
      --hex[=false]                           print byte strings as hex encoded strings
      --insecure-skip-tls-verify[=false]      skip server certificate verification
      --insecure-transport[=true]             disable transport security for client connections
      --key=""                                identify secure client using this TLS key file
      --user=""                               username[:password] for authentication (prompt if password is not supplied)
  -w, --write-out="simple"                    set the output format (fields, json, protobuf, simple, table)

root@master:/home/ubuntu#
```

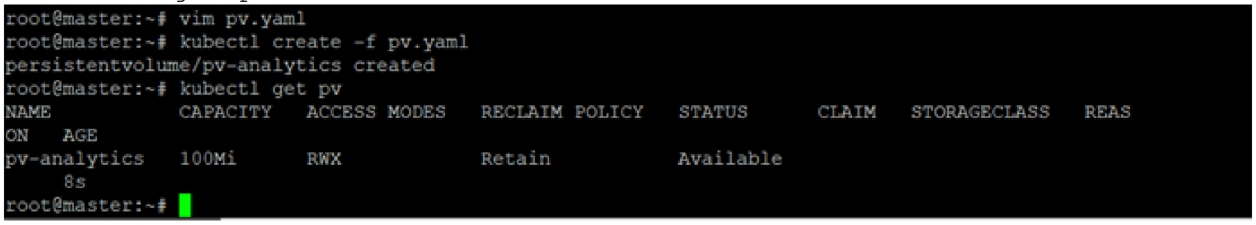
```
$ ETCDCTL_API=3 etcdctl snapshot restore /opt/etcd-snapshot.db --
endpoints=https://127.0.0.1:2379 --
cert=/etc/kubernetes/pki/etcd/server.crt --
cacert=/etc/kubernetes/pki/etcd/ca.crt --
key=/etc/kubernetes/pki/etcd/server.key --data-dir=/var/lib/etcd --
initial-advertise-peer-urls=http://10.0.0.4:2380 --initial-
cluster=<master-name>=http://10.0.0.4:2380" --initial-cluster-token="etcd-
cluster" --name="<master-name>"
```

Q4) Create a Persistent Volume with the given specification.

Volume Name: pv-analytics, Storage: 100Mi, Access modes: ReadWriteMany, Host Path: /pv/data-analytics

Ans.

```
$ vim pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-analytics
spec:
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteMany
  hostPath:
    path:  /pv/data-analytics
$ kubectl create -f pv.yaml
$ kubectl get pv
```



```
root@master:~# vim pv.yaml
root@master:~# kubectl create -f pv.yaml
persistentvolume/pv-analytics created
root@master:~# kubectl get pv
NAME           CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REAS
ON   AGE
pv-analytics   100Mi      RWX            Retain           Available
     8s
root@master:~#
```

Read More: K8s Persistent Storage

Q5) Taint the worker node to be Unschedulable. Once done, create a pod called dev-redis, image redis:alpine to ensure workloads are not scheduled to this worker node. Finally, create a new pod called prod-redis and image redis:alpine with toleration to be scheduled on node01.

key:env_type, value:production, operator: Equal and effect:NoSchedule

Ans.

```
$ kubectl get nodes
$ kubectl taint node node01 env_type=production:NoSchedule
$ kubectl describe nodes node01 | grep -i taint
$ kubectl run dev-redis --image=redis:alpine --dyn-run=client -o yaml >
pod-redis.yaml
$ vi prod-redis.yaml
apiVersion: v1
kind: Pod
metadata:
  name: prod-redis
spec:
  containers:
  - name:  prod-redis
    image:  redis:alpine
  tolerations:
  - effect: Noschedule
    key: env_type
    operator: Equal
    value: prodcution
$ kubectl create -f prod-redis.yaml
```

Read More: Scheduling in K8s

Q6) Set the node named worker node as unavailable and reschedule all the pods running on it. *(Drain node)*

Ans.

```
$ Kubectl drain node <worker node> --ignore-daemonsets
```

Q7) Create a Pod called non-root-pod , image: redis:alpine

runAsUser: 1000

fsGroup: 2000

Ans.

```
$ vim non-root-pod.yaml
$ kubectl create -f non-root-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name:  non-root-pod
spec:
  securityContext:
    runAsUser:  1000
    fsGroup:  2000
  containers:
  -  name:  non-root-pod
```

Read More: K8s Pods For Beginners

Q8) Create a NetworkPolicy which denies all ingress traffic

Ans.

```
$ vim policy.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector: {}
  policyTypes:
  - Ingress
$ kubectl create -f policy.yaml
```

Read More: K8s Network Policy

Conclusion

Kubernetes is the leading technology, and companies always look for skilled employees. To help you crack the CKA exam and secure a job, we put some effort and listed some Sample Exam Questions.