CKA Mock Exam - 2

Take a backup of the etcd cluster and save it to /tmp/etcd-backup.db

check the etcd version

```
ETCDCTL_API=3 etcdctl version
```

chek the endpoint

```
cat /etc/kubernetes/manifests/etcd.yaml | grep advertise-client-urls
```

check the etcd command working

```
ETCDCTL_API=3 etcdctl \
  --endpoints=https://172.17.0.41:2379 \
  --cacert="/etc/kubernetes/pki/etcd/ca.crt" \
  --cert="/etc/kubernetes/pki/etcd/server.crt" \
  --key="/etc/kubernetes/pki/etcd/server.key" \
  member list
```

take backup

```
ETCDCTL_API=3 etcdctl \
  --endpoints=https://172.17.0.41:2379 \
  --cacert="/etc/kubernetes/pki/etcd/ca.crt" \
  --cert="/etc/kubernetes/pki/etcd/server.crt" \
  --key="/etc/kubernetes/pki/etcd/server.key" \
  snapshot save /tmp/etcd-backup.db
```

verify backup

```
ETCDCTL_API=3 etcdctl \
  --endpoints=https://172.17.0.41:2379 \
  --cacert="/etc/kubernetes/pki/etcd/ca.crt" \
  --cert="/etc/kubernetes/pki/etcd/server.crt" \
  --key="/etc/kubernetes/pki/etcd/server.key" \
  snapshot status /tmp/etcd-backup.db -w table
```

Create a Pod called redis-storage with image: redis:alpine with a Volume of type emptyDir that lasts for the life of the Pod. Specs on the right.

```
kubectl run redis-storage --restart=Never --image=redis:alpine --dry-run -o yaml

cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: redis-storage
  name: redis-storage
spec:
  containers:
  - image: redis:alpine
    name: redis-storage
    resources: {}
    volumeMounts:
    - mountPath: /data/redis
      name: redis-storage
  volumes:
  - name: redis-storage
    emptyDir: {}
EOF
```

Create a new pod called super-user-pod with image busybox:1.28. Allow the pod to be able to set system_time

```
kubectl run super-user-pod --restart=Never --image=busybox:1.28 --dry-run -o yaml --command -- sleep 4800

cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: super-user-pod
  name: super-user-pod
spec:
  containers:
  - command:
    - sleep
    - "4800"
    image: busybox:1.28
    securityContext:
      capabilities:
        add: ["SYS_TIME"]
    name: super-user-pod
EOF
```

A pod definition file is created at /root/use-pv.yaml. Make use of this manifest file and mount the persistent volume called pv-1. Ensure the pod is running and the PV is bound.

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Mi
EOF

cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: use-pv
  name: use-pv
spec:
  containers:
  - image: nginx
    name: use-pv
    resources: {}
    volumeMounts:
    - mountPath: "/data"
      name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: my-pvc
EOF
```

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.

```
kubectl run nginx-deploy --image=nginx:1.16 --record

kubectl set image deploy nginx-deploy nginx-deploy=nginx:1.17 --record
```

Create a new user called john. Grant him access to the cluster. John should have permission to create, list, get, update and delete pods in the development namespace . The private key exists in the location: /root/john.key and csr at /root/john.csr

```
cat << EOF | kubectl apply -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: john-developer
spec:
  request: $(cat /root/john.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - server auth
EOF

kubectl certificate approve john-developer

kubectl create role developer --verb=create,list,get,update,delete --resource=pods --namespace=development

kubectl create rolebinding developer-role-binding --user=john --role=developer --namespace=development
```

test

```
kubectl auth can-i update pods --as john -n development
```

Create an nginx pod called nginx-resolver using image nginx, expose it internally with a service called nginx-resolver-service. Test that you are able to look up the service and pod names from within the cluster. Use the image: busybox:1.28 for dns lookup. Record results in /root/nginx.svc and /root/nginx.pod

```
kubectl run nginx-resolver --restart=Never --image=nginx

kubectl expose pod nginx-resolver --name=nginx-resolver-service --port=80

kubectl run test-nslookup --restart=Never --image=busybox:1.28 --rm -it -- nslookup nginx-resolver-service > /root/nginx.svc

kubectl run test-nslookup --restart=Never --image=busybox:1.28 --rm -it -- nslookup 10-32-0-6.default.pod > /root/nginx.pod
```

update the IP before running the command

Create a static pod on node01 called nginx-critical with image nginx. Create this pod on node01 and make sure that it is recreated/restarted automatically in case of a failure.

check the kubelet status

```
systemctl status kubelet
```

check for the following config

```
--config=/var/lib/kubelet/config.yaml
```

get static pod path location

```
cat /var/lib/kubelet/config.yaml | grep staticPodPath
```

create the folder

```
mkdir -p /etc/kubernetes/manifests && cd /etc/kubernetes/manifests
```

past this in `nginx-critical.yaml` on node01

```
kubectl run nginx-critical --restart=Never --image=nginx --dry-run -o yaml
```