



Kubernetes CKA/D Sample Exam Questions

Atul Kumar

Author & Cloud Expert



[EDITION 01]

The **Certified Kubernetes Administrator (CKA)** and **Certified Kubernetes Application Developer (CKAD)** exams are both hands-on, performance-based assessments. These exams are conducted by the **Cloud Native Computing Foundation (CNCF)** and are designed to evaluate the practical skills and knowledge of individuals working with Kubernetes.

Here's a brief overview of each exam:

Certified Kubernetes Administrator (CKA):



Certified Kubernetes Administrator certification is to provide assurance that Kubernetes Administrators have the skills, knowledge, to perform the responsibilities of **Kubernetes administrators**.

Know more about [Certified Kubernetes Administrator](#)

Certified Kubernetes Application Developer (CKAD):



The **Certified Kubernetes Application Developer** certification is designed to guarantee that certification holders have the knowledge, skills, and capability to design, configure, and expose cloud-native applications for Kubernetes and also perform the responsibilities of Kubernetes application developers. Hence, it also assures that the Kubernetes Application Developer can use core primitives to build, monitor, and troubleshoot scalable applications in Kubernetes.

Know more about [CKAD](#)

Q1) Create a new Deployment with 2 replicas and use busybox image.

Ans:

```
kubectl create deployment test-deployment --image=busybox --replicas=2
```

or

```
vim test-deployment.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      name: busybox-pod
  template:
    metadata:
      labels:
        name: busybox-pod
    spec:
      containers:
        - name: busybox-container
          image: busybox
```

```
kubectl create -f test-deployment.yaml
```

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 a new service “web-application” .

Name: web-application; Type: NodePort; ; port: 8080; nodePort: 30083; selector: simple-webapp

Ans:

```
vim web-application.yaml
```

```
apiVersion: v1
kind: Service
metadata:
  name: web-application
```

```
spec:
  type: NodePort
  ports:
    - targetPort: 8080
      port: 8080
      nodePort: 30083
  selector:
    name: simple-webapp
```

```
kubectcl create -f web-application.yaml
```

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
```

```
kubectcl create -f pv.yaml
```

```
kubectcl get pv
```

```
root@master:~# vim pv.yaml
root@master:~# kubectcl create -f pv.yaml
persistentvolume/pv-analytics created
root@master:~# kubectcl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REAS
pv-analytics	100Mi	RWX	Retain	Available			

```
root@master:~#
```

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
```

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

Ans:

```
Kubectrl 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
```

Q8) 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:

```
kubectl create serviceaccount pvviewer  
kubectl create clusterrole pvviewer-role --resource=persistentvolumes --verb=list  
kubectl create clusterrolebinding pvviewer-role-binding --clusterrole=pvviewer-role --  
serviceaccount=default:pvviewer
```

```
apiVersion: v1  
kind: Pod  
metadata:  
  name: pvviewer  
spec:  
  containers:  
  - image: redis  
    name: pvviewer  
  serviceAccountName: pvviewer
```

```
kubectl create -f pvviewer.yaml
```



```

root@master: ~
root@master:~# kubectl create serviceaccount pvviewer
serviceaccount/pvviewer created
root@master:~# kubectl create clusterrole pvviewer-role --resource=persistentvolumes --verb=list
clusterrole.rbac.authorization.k8s.io/pvviewer-role created
root@master:~# kubectl create clusterrolebinding pvviewer-role-binding --clusterrole=pvviewer-role --serviceaccount=default:pvviewer
clusterrolebinding.rbac.authorization.k8s.io/pvviewer-role-binding created
root@master:~# vim pv.yaml
root@master:~# vim pvviewer.yaml
root@master:~# kubectl create -f pvviewer.yaml
pod/pvviewer created
root@master:~#

```

```

root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe pod pvviewer
Name:          pvviewer
Namespace:     default
Priority:       0
Node:          worker2/10.0.0.6
Start Time:    Mon, 21 Sep 2020 06:40:06 +0000
Labels:        <none>
Annotations:   <none>
Status:        Running
IP:            10.32.0.2
IPs:
  IP: 10.32.0.2
Containers:
  pvviewer:
    Container ID:  docker://01e73e0536affa5c0ce12505d3379f071d4a3c2d6d22b894b8776899a745bafc
    Image:         redis
    Image ID:      docker-pullable://redis@sha256:1cfb205a988a9dae5f025c57b92e9643ec0e7ccff6e66bc639d8a5f95bba928c
    Port:         <none>
    Host Port:    <none>
    State:        Running
      Started:    Mon, 21 Sep 2020 06:40:10 +0000
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from pvviewer-token-h974d (ro)
Conditions:
  Type             Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  pvviewer-token-h974d:
    Type:          Secret (a volume populated by a Secret)
    SecretName:     pvviewer-token-h974d
    Optional:       false
QoS Class:         BestEffort
Node-Selectors:    <none>
Tolerations:       node.kubernetes.io/not-ready:NoExecute for 300s
                   node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type     Reason         Age   From          Message
  ----     -
  Normal   Scheduled      39s   default-scheduler   Successfully assigned default/pvviewer to worker2
  Normal   Pulling        38s   kubelet, worker2    Pulling image "redis"
  Normal   Pulled         36s   kubelet, worker2    Successfully pulled image "redis"

```

Q9) 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
kubectrl create -f policy.yaml
```

Q10) Create a pod myapp-pod and that use an initContainer that uses the busybox image and sleeps for 20 seconds.

Ans:

```
vim myapp.yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
    - name: myapp-container
      image: busybox:1.28
      command: ['sh', '-c', 'echo The app is running! && sleep 3600']
  initContainers:
    - name: init-myservice
      image: busybox
      command: ["sleep", "20"]
kubectrl create -f myapp.yaml
```

Q11) Create the ingress resource with name ingress-wear-watch to make the applications available at /wear on the Ingress service in app-space namespace.

Ans:

```
vim ingress.yaml

apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-wear-watch
  namespace: app-space
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /wear
        backend:
          serviceName: wear-service
          servicePort: 8080

kubectl create -f ingress.yaml
```

Verify//

```
kubectl describe ingress ingress-wear-watch -n app-space

Name:          ingress-wear-watch
Namespace:     app-space
Address:
Default backend: default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
Rules:
  Host      Path  Backends
  ----      -
  *
            /wear wear-service:8080 10.244.1.2:8080)
Annotations:  nginx.ingress.kubernetes.io/rewrite-target: /
Events:       <none>
```

Q12) Schedule pod for node

```
Name: nginx
image: nginx
Node Selector: disk=ssd
```

Ans:

search: node selector

<https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/>

```
vim nodeslector.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
  nodeSelector:
    disk: ssd
```

```
kubectl create -f nodeslector.yaml
```

Q13) Create a new pod called **super-user-pod** with image **busybox:1.28**. Allow the pod to be able to set `system_time`. **The container should sleep for 4800 seconds.**

Ans:

```
vim super-user-pod.yaml
```

```
kubectl create -f super-user-pod.yaml
```

```
root@master:~# vim super-user-pod.yaml
root@master:~# kubectl create -f super-user-pod.yaml
pod/super-user-pod created
root@master:~# █
```

```
root@master: ~  
apiVersion: v1  
kind: Pod  
metadata:  
  name: super-user-pod  
spec:  
  containers:  
  - name: super-user-pod  
    image: busybox:1.28  
    command: ["sleep", "4800"]  
    securityContext:  
      capabilities:  
        add: ["SYS_TIME"]  
~  
~  
~  
~  
~
```

Get a shell into the running Container:

```
kubectl exec -it super-user-pod -- sh
```

In your shell, view the capabilities for process 1:

```
cd /proc/1  
cat status
```

Check more on: <https://kubernetes.io/docs/tasks/configure-pod-container/security-context/>

Q14) Remove taint from the master node and verify node is untaint.

Ans:

```
kubectl taint node master node-role.kubernetes.io/master-  
kubectl describe nodes | egrep "Name:|Taints:"
```

Q15) Create a configmap called myconfigmap with literal value appname=myapp

Ans:

```
kubectl create cm myconfigmap --from-literal=appname=myapp
```

FREE Consultation Call

Whether you're struggling to get a job or confused about where to even begin to get a thriving career, book a free clarity call with one of our experts at the time that works best for you.

BOOK A CALL NOW



THANK YOU

www.k21academy.com