# Certified Kubernetes Administrator Certification CKA Exam Dumps

Certified Kubernetes Administrator certification CKA exam dumps are released today. We all know that the Certified Kubernetes Administrator (CKA) program was created by the Cloud Native Computing Foundation (CNCF), in collaboration with The Linux Foundation, to help develop the Kubernetes ecosystem. CKA exam dumps from DumpsBase are great to help you practice and prepare for the Certified Kubernetes Administrator exam.

## Check CKA Certified Kubernetes Administrator Free Dumps

Top of Form

**1. Create a namespace called 'development' and a pod with image nginx called nginx on this namespace.**

Solution: kubectl create namespace development kubectl run nginx --image=nginx --restart=Never -n development

**2. Create a nginx pod with label env=test in engineering namespace See the solution below.**

Solution: kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry- run -o yaml > nginx-pod.yaml kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry- run -o yaml | kubectl create -n engineering -f - YAML File: apiVersion: v1 kind: Pod metadata: name: nginx namespace: engineering labels: env: test spec: containers: - name: nginx image: nginx imagePullPolicy: IfNotPresent restartPolicy: Never kubectl create -f nginx-pod.yaml

**3. Get list of all pods in all namespaces and write it to file "/opt/pods-list.yaml"**

Solution: kubectl get po -all-namespaces > /opt/pods-list.yaml

**4. Create a pod with image nginx called nginx and allow traffic on port 80**

Solution: kubectl run nginx --image=nginx --restart=Never --port=80

**5. Create a busybox pod that runs the command "env" and save the output to "envpod" file**

Solution: kubectl run busybox --image=busybox --restart=Never --rm -it -- env > envpod.yaml

**6. List pod logs named "frontend" and search for the pattern "started" and write it to a file "/opt/error- logs"**

Solution: Kubectl logs frontend | grep -i "started" > /opt/error-logs

**7. Create a pod that echo "hello world" and then exists. Have the pod deleted automatically when it's completed**

Solution: kubectl run busybox --image=busybox -it --rm --restart=Never -- /bin/sh -c 'echo hello world' kubectl get po # You shouldn't see pod with the name "busybox"

**8. Create a pod with environment variables as var1=value1.Check the environment variable in pod**

Solution: kubectl run nginx --image=nginx --restart=Never --env=var1=value1 # then kubectl exec -it nginx -- env # or kubectl exec -it nginx -- sh -c 'echo $var1' # or kubectl describe po nginx | grep value1

**9. Get list of all the pods showing name and namespace with a jsonpath expression.**

Solution: kubectl get pods -o=jsonpath="{.items[\*]['metadata.name' , 'metadata.namespace']}"

**10. Check the image version in pod without the describe command**

Solution: kubectl get po nginx -o jsonpath='{.spec.containers[].image}{"  
"}'

**11. List the nginx pod with custom columns POD\_NAME and POD\_STATUS**

Solution: kubectl get po -o=custom-columns="POD\_NAME:.metadata.name, POD\_STATUS:.status.containerStatuses[].state"

**12. List all the pods sorted by name**

Solution: kubect1 get pods --sort-by=.metadata.name

**13. Create a pod that having 3 containers in it? (Multi-Container)**

Solution: image=nginx, image=redis, image=consul Name nginx container as "nginx-container" Name redis container as "redis-container" Name consul container as "consul-container" Create a pod manifest file for a container and append container section for rest of the images kubectl run multi-container --generator=run-pod/v1 --image=nginx -- dry-run -o yaml > multi- container.yaml # then vim multi-container.yaml apiVersion: v1 kind: Pod metadata: labels: run: multi-container name: multi-container spec: containers: - image: nginx name: nginx-container - image: redis name: redis-container - image: consul name: consul-container restartPolicy: Always

**14. Create 2 nginx image pods in which one of them is labelled with env=prod and another one labelled with env=dev and verify the same.**

Solution: kubectl run --generator=run-pod/v1 --image=nginx -- labels=env=prod nginx-prod --dry-run -o yaml > nginx-prodpod.yaml Now, edit nginx-prod-pod.yaml file and remove entries like "creationTimestamp: null" "dnsPolicy: ClusterFirst" vim nginx-prod-pod.yaml apiVersion: v1 kind: Pod metadata: labels: env: prod name: nginx-prod spec: containers: - image: nginx name: nginx-prod restartPolicy: Always # kubectl create -f nginx-prod-pod.yaml kubectl run --generator=run-pod/v1 --image=nginx -- labels=env=dev nginx-dev --dry-run -o yaml > nginx-dev-pod.yaml apiVersion: v1 kind: Pod metadata: labels: env: dev name: nginx-dev spec: containers: - image: nginx name: nginx-dev restartPolicy: Always # kubectl create -f nginx-prod-dev.yaml Verify : kubectl get po --show-labels kubectl get po -l env=prod kubectl get po -l env=dev

**15. Get IP address of the pod - "nginx-dev"**

Solution: Kubect1 get po -o wide Using JsonPath kubect1 get pods -o=jsonpath='{range .items[\*]}{.metadata.name}{"t"}{.status.podIP}{"  
"}{end}'

**16. Print pod name and start time to "/opt/pod-status" file**

Solution: kubect1 get pods -o=jsonpath='{range .items[\*]}{.metadata.name}{"t"}{.status.podIP}{"  
"}{end}'

**17. Check the Image version of nginx-dev pod using jsonpath**

Solution: kubect1 get po nginx-dev -o jsonpath='{.spec.containers[].image}{"  
"}'

**18. Create a busybox pod and add "sleep 3600" command**

Solution: kubectl run busybox --image=busybox --restart=Never -- /bin/sh -c "sleep 3600"

**19. Create an nginx pod and list the pod with different levels of verbosity**

Solution: // create a pod kubectl run nginx --image=nginx --restart=Never --port=80 // List the pod with different verbosity kubectl get po nginx --v=7 kubectl get po nginx --v=8 kubectl get po nginx --v=9

**20. List the nginx pod with custom columns POD\_NAME and POD\_STATUS**

Solution: kubectl get po -o=custom-columns="POD\_NAME:.metadata.name, POD\_STATUS:.status.containerStatuses[].state"

**21. List all the pods sorted by name**

Solution: kubectl get pods --sort-by=.metadata.name

**22. List all the pods sorted by created timestamp**

Solution: kubect1 get pods--sort-by=.metadata.creationTimestamp

**23. List all the pods showing name and namespace with a json path expression**

Solution: kubectl get pods -o=jsonpath="{.items[\*]['metadata.name', 'metadata.namespace']}"

**24. List "nginx-dev" and "nginx-prod" pod and delete those pods**

Solution: kubect1 get pods -o wide kubectl delete po "nginx-dev" kubectl delete po "nginx-prod"

**25. Delete the pod without any delay (force delete)**

Solution: Kubect1 delete po "POD-NAME" --grace-period=0 -force

**26. Create a redis pod and expose it on port 6379**

Solution: kubectl run redis --image=redis --restart=Never --port=6379 YAML File : apiVersion: v1 kind: Pod metadata: labels: run: redis name: redis spec: containers: - image: redis name: redis ports: - containerPort: 6379 Rt restartPolicy: Always

**27. Create the nginx pod with version 1.17.4 and expose it on port 80**

Solution: kubectl run nginx --image=nginx:1.17.4 --restart=Never -- port=80

**28. Change the Image version to 1.15-alpine for the pod you just created and verify the image version is updated.**

Solution: Kubect1 set image pod/nginx nginx=nginx:1.15-alpine kubect1 describe po nginx // another way it will open vi editor and change the version kubeclt edit po nginx kubect1 describe po nginx

**29. Change the Image version back to 1.17.1 for the pod you just updated and observe the changes**

Solution: kubectl set image pod/nginx nginx=nginx:1.17.1 kubectl describe po nginx kubectl get po nginx -w # watch it

**30. Create a redis pod, and have it use a non-persistent storage Note: In exam, you will have access to kubernetes.io site, Refer : https://kubernetes.io/docs/tasks/configure-pod-container/configurevolume-storage/**

Solution: apiVersion: v1 kind: Pod metadata: name: redis spec: containers: - name: redis image: redis volumeMounts: - name: redis-storage mountPath: /data/redis ports: - containerPort: 6379 volumes: - name: redis-storage emptyDir: {}

**31. Create a Pod with three busy box containers with commands "ls; sleep 3600;", "echo Hello World; sleep 3600;" and "echo this is the third container; sleep 3600" respectively and check the status**

Solution: // first create single container pod with dry run flag kubectl run busybox --image=busybox --restart=Always --dry-run -o yaml -- bin/sh -c "sleep 3600; ls" > multi-container.yaml // edit the pod to following yaml and create it apiVersion: v1 kind: Pod metadata: labels: run: busybox name: busybox spec: containers: - args: - bin/sh - -c - ls; sleep 3600 image: busybox name: busybox-container-1 - args: - bin/sh - -c - echo Hello world; sleep 3600 image: busybox name: busybox-container-2 - args: - bin/sh - -c - echo this is third container; sleep 3600 image: busybox name: busybox-container-3 restartPolicy: Always // Verify Kubectl get pods

**32. Check logs of each container that "busyboxpod-{1,2,3}"**

Solution: kubectl logs busybox -c busybox-container-1 kubectl logs busybox -c busybox-container-2 kubectl logs busybox -c busybox-container-3

**33. Create a Pod with main container busybox and which executes this "while true; do echo `Hi I am from Main container' >> /var/log/index.html; sleep 5; done" and with sidecar container with nginx image which exposes on port 80. Use emptyDir Volume and mount this volume on path /var/log for busybox and on path /usr/share/nginx/html for nginx container. Verify both containers are running.**

Solution: // create an initial yaml file with this kubectl run multi-cont-pod --image=busbox --restart=Never -- dry-run -o yaml > multi-container.yaml // edit the yml as below and create it kubectl create -f multi-container.yaml vim multi-container.yaml apiVersion: v1 kind: Pod metadata: labels: run: multi-cont-pod name: multi-cont-pod spec: volumes: - name: var-logs emptyDir: {} containers: - image: busybox command: ["/bin/sh"] args: ["-c", "while true; do echo 'Hi I am from Main container' >> /var/log/index.html; sleep 5;done"] name: main-container volumeMounts: - name: var-logs mountPath: /var/log - image: nginx name: sidecar-container ports: - containerPort: 80 volumeMounts: - name: var-logs mountPath: /usr/share/nginx/html restartPolicy: Never // Create Pod kubectl apply -f multi-container.yaml //Verify kubectl get pods

**34. Exec into both containers and verify that main.txt exist and**

Solution: NEED TO WRITE ANSWER FOR THIS

**35. Create an nginx pod and set an env value as 'var1=val1'. Check the env value existence within the pod**

Solution: kubectl run nginx --image=nginx --restart=Never --env=var1=val1 # then kubectl exec -it nginx -- env # or kubectl exec -it nginx -- sh -c 'echo $var1' # or kubectl describe po nginx | grep val1 # or kubectl run nginx --restart=Never --image=nginx --env=var1=val1 -it --rm - env

Bottom of Form