**Certified Kubernetes Administrator Lab Questions**

**Q.1 Create a new pod called “my-pod01” with image “busybox” and allow the pod to be able to set system\_time. The container should sleep for 3200 seconds.**

**Solution:**

#kubectl run mypod01 --image=busybox --command sleep 3200 --dry-run=client -o yaml

# Create pod YAML like below:

***apiVersion: v1***

***kind: Pod***

***metadata:***

***labels:***

***run: mypod01***

***name: mypod01***

***spec:***

***containers:***

***- command:***

***- sleep***

***- "3200"***

***image: busybox***

***name: mypod01***

***securityContext:***

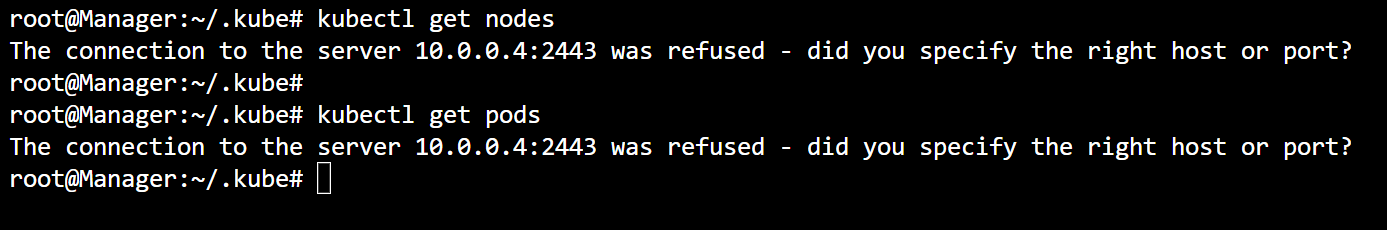
***capabilities:***

***add: ["SYS\_TIME"]***

# kubectl create -f mypod.yaml

# kubectl get pod

**Q.2 A kubeconfig file called “config” is placed in /root/.kube/config. There is something wrong with the configuration file. How you can troubleshoot the below issue?**

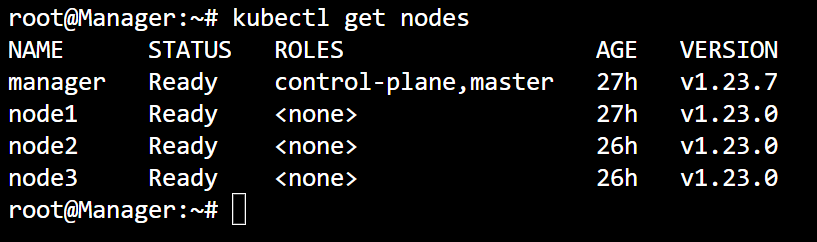


Solution:

# kubectl config view “API-Server Port is different”

# vim .kube/config

# kubectl config view



**Q.3 Create a new deployment named as web-proj-10 with image nginx:1.16 with one replica. After that upgrade the deployment to version nginx:1.17 using rolling update policy. Make sure the version upgrade recorded in the resource annotation.**

Solution:

# kubectl create deployment web-proj-10 --image=nginx:1.16

# kubectl get deployments.apps

# kubectl get pods

# kubectl set image deployment/web-proj-10 nginx=nginx:1.17 –record

# kubectl get pods

# kubectl rollout history deployment web-proj-10

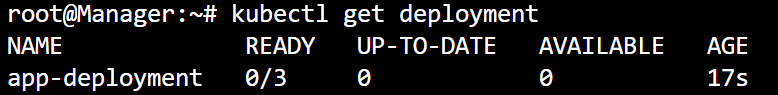
# kubectl describe pod web-proj-10-7d6cb7fd75-fvmq9

**Q.4 Create a new deployment called “app-deployment”, scale the deployment to 3 replicas. Make sure desired no. of pod always running.**

Solution:

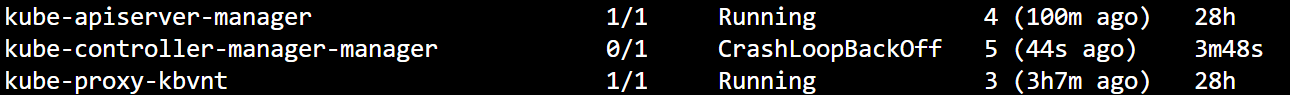
# kubectl create deployment app-deployment --image=nginx --replicas=3

# Kubectl get deployment



**Note:** Your deployment got created but not ready with three replicas.

# kubectl get pods -n kube-system

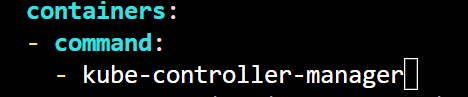


Note: some problem with kube-controller manager

# # kubectl describe pod -n kube-system kube-controller-manager-manager

Note: refer the even logs

# vim /etc/kubernetes/manifests/kube-controller-manager.yaml



Note: keep it as like: kube-controller-manager and wait for 2 min.

# kubectl get deployments.apps

**Q.5 Upgrade your Kubernetes Cluster (Master and Worker Node) from 1.22.0 to 1.23.7 version. Drain your both Master and Worker node, make it ready for upgrade. Make sure Master and worker node must be ready after upgrade.**

Solution: Take reference from Kubernetes upgrade module.

**Q.6 Create a static pod named as static-pod with image nginx on node1. You have to make sure that it is recreated/restarted automatically in case of any failure.**

Solution: Take reference from POD&QoS Module.

**Q.7 Create a pod named as “multi-pod” with two containers as mentioned below requirement:**

**Container-1: Name “web” image “nginx”.**

**Container-2: Name “busybox” image “busybox” commad “sleep 4800”.**

Solution:

1. Create a multi-container pod using below yaml.

***apiVersion: v1***

***kind: Pod***

***metadata:***

***name: multi-pod***

***labels:***

***app: web***

***spec:***

***containers:***

***- name: web***

***image: nginx***

***ports:***

***- containerPort: 80***

***- name: busybox***

***image: busybox***

***command: ["sleep","4800"]***

# kubectl create -f multi-pod.yaml

# kubectl describe pod multi-pod

**Q.8 Create a pod named as env-pod in “dev” namespace belonging to the development environment (env=dev) and frontend tier (tier=front).**

**Container image should be nginx:1.16.**

Solution:

# kubectl create ns dev

# kubectl get ns

# kubectl run env-pod --image=nginx:1.16 -n dev --labels env=dev,tier=front

# kubectl get pod -n dev

# kubectl get pod -n dev --show-labels

**Q.9 Get the node “Node01” in JSON format and store it a file at /var/node01.json.**

Solution:

# kubectl get nodes

# kubectl get nodes node1 -o json

# kubectl get nodes node1 -o json > /var/node01.json

**Q.10 Take a backup of ETCD database save it to “/root/” with name of backup “etcd-backup.db”.**

Solution: Take a reference for more info Kubernetes upgrade module.

# kubectl get pod -n kube-system

# kubectl -n kube-system describe pod etcd-manager

# apt install etcd-client

# ETCDCTL\_API=3 etcdctl --endpoints=https://127.0.0.1:2379 \

--cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key \

snapshot save etcd-backup.db

**Q.11 Create a pod called web-pod using image nginx, expose it internally with a service called web-pod-svc. Check that you are able to lookup the service and pod within the cluster. Use the image “busybox” for dns lookup.**

Solution:

# kubectl run web-pod --image=nginx

# kubectl get pod

# kubectl expose pod web-pod --name=web-pod-svc --port=80

# kubectl get service

For testing create busybox pod

# kubectl run nslookup-pod --image=busybox:1.28 --command sleep 4800

**Q.12 Use JSON path query to retrieve the osImages of all the nodes and store it in a file “all\_node\_osimage.txt” at “/root” location.**

Solution:

# kubectl get nodes

# kubectl get nodes -o json

# kubectl get nodes -o jsonpath='{.items[\*].status.nodeInfo.osImage}'

# echo $(kubectl get nodes -o jsonpath='{.items[\*].status.nodeInfo.osImage}') > /root/all\_node\_osimage.txt

**Q.13 Create a Persistent Volume(PV) with given specification:**

**PV Name: my-pv**

**Storage: 100Mi**

**Access Mode: ReadWriteMany**

**HostPath: /my-pv/data**

Solution: Take a reference from Kubernetes Storage Module.

**Q.14 Create a pod named as my-web-pod with image “ankitsdocker/images:customweb” and expose this pod as service named as “my-svc” type nodeport with port number “32002”.**

**Note: You web application is listing on port no. 81.**

Solution:

# kubectl run my-web-pod --image=ankitsdocker/images:customweb

# kubectl get pod

# kubectl expose pod my-web-pod --type=NodePort --port=32002 --target-port=81 --name=my-svc -o yaml

Note: edit the required field or can take reference from Kubernetes Module.

# kubectl get service

# kubectl describe service my-svc

**Q.15 Taint the worker node “Node1” with details provided below:**

**Key: env\_type, value:production, operator: Equal and effect: NoSchedule**

**Create a pod called dev-pod-nginx using image=nginx, make sure that workloads are not scheduled to this worker node “Node1”.**

**Create an another pod called prod-pod-nginx using image=nginx with toleration to be scheduled on “Node1”.**

Solution: Take a reference from Kubernetes NodeAffinity and Toleration Module.

**Q.16 Create a pod called pod-sec with given specification:**

**1. Security Context: runAsUser:1000, fsGroup:2000**

**2. Image: redis:alpine**

Solution:

***apiVersion: v1***

***kind: Pod***

***metadata:***

***name: pod-sec***

***labels:***

***env: sec***

***spec:***

***securityContext:***

***runAsUser: 1000***

***fsGroup: 2000***

***containers:***

***- name: sec-con***

***image: redis:alpine***

# kubectl exec -it pod-sec – whoami

**Q.17 Worker node “Node1” is not responding, have a look a fix the issue.**

**Note: On worker Node1 kubelet service was not running.**

Solution:

# systemctl start kubelet

**Q.18 Create a static pod called “web-static” with image “busybox” on control plane node. Move the static pod to worker node “Node1”.**

Solution:

For creating static pod- Take a reference from POD&QoS Module.

For Moving static pod- just move the static pod definition from control plan /etc/kubernetes/manifests to worker node /etc/kubernetes/manife location.

**Q.19 Create a new user named as “alok”. Grant him access to the cluster. User “alok” should have permission to create, list, get, update and delete pods in “space” namespace.**

**The private key exists at location: /root/alok.key and csr /root/alok.csr.**

Solution: Need to work on it.

**Q.20 Create a PV, PVC and POD with below specification:**

**1. PV**

**Volume Name: mypvlog**

**Storage: 100Mi**

**Access Mode: ReadWriteMany**

**Host Path: /pv/log**

**Reclaim Policy: Retain**

**PVC**

**Volume Name: pv-claim-log**

**Storage Request: 50 Mi**

**Access Mode: ReadWriteMany**

**POD**

**Name: storage-pod**

**Image Name: nginx**

**PVC: “pv-claim-log”**

**Volume Mount: /log**

Solution: Take a reference from Kubernetes Storage Module.

**Q.21 Worker node “Node1” is not responding, have look and fix the issue.**

RCA: Kubelet was not running properly. Status was “in-active” (autorestart).

Log: “Failed to construct kubelet dependencies" err="unable to load client CA file /etc/kubernetes/pki/abc.crt”

After seeing the log I identified the certificate name was incorrect in file:

Vim /var/lib/kubelet/config.yaml

Change the cert value as

x509:

clientCAFile: /etc/kubernetes/pki/<give the correct key>

**Q.22 A pod named as “my-pod” in default namespace is not running state. Find the problem and make it running.**

Solution: Describe the pod and read the events and log.

**Q.23 Create a new deployment called “nginx-deployment” using image “nginx” along with 6 replicas. There are 3 worker node (Node1, Node2 and Node3) in cluster. Make sure the pod should not be scheduled on “Node3”.**

**Note: Revert the changes that you did in the cluster.**

Solution:

Method-1: Cordon the node3

# kubectl cordon Node3

Method-2: Taint the node3

**Q.24 Create a Replica set (web-pod, image: nginx, Replica:3), There is one pod named as “web-pod” already running in cluster.**

**Make sure the total number of running pod should not be more than 3.**

Solution:

Create a replica set with label as same as already running pod web-pod with two replicas only.

**Q.25 There are three node is cluster. Create a DaemonSet(Name: my-pod, image: nginx) on each node except on Node3.**

Solution: Taint the Node three with any key:value

**Q.26 Create a pod named as “tmp-pod” with image “busybox” that should wire “Binary has been downloaded” into file named as “log.txt”. Make sure pod should be deleted automatically when it’s completed.**

Solution:

# kubectl run tmp-pod --image=busybox -it --rm --restart=Never -- /bin/sh -c 'echo Binary has been Downloaded' > log.txt

# cat log.txt