

**Linux Foundation**

**CKA Exam**

**Certified Kubernetes Administrator (CKA) Program**

**Version : 8.0**

**[Total Questions : 67]**

## Question: 1

Task Weight: 4%

Set configuration context:

```
[student@node-1] $ kubectl config use-context k8s
```

Task

Schedule a Pod as follows:

- \* Name: kucc1
- \* App Containers: 2
  - o nginx
  - o consul
- \* Container Name/Images:

A. Explanation:

Solution:

```
student@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
student@node-1:~$ kubectl run kucc1 --image=nginx --dry-run=client -o yaml > aa.y
```

Readme Web Terminal THE LINUX FOUNDATION

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: kucc1
    name: kucc1
spec:
  containers:
  - image: nginx
    name: nginx
  - image: consul
    name: consul
```

```
student@node-1:~$ kubectl config use-context k8s
switched to context "k8s".
student@node-1:~$ kubectl run kucc1 --image=nginx --dry-run=client -o yaml > aa.yaml
student@node-1:~$ vim aa.yaml
student@node-1:~$ kubectl create -f aa.yaml
pod/kucc1 created
student@node-1:~$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
11-factor-app 1/1     Running   0          6h34m
cpu-loader-98b9se 1/1     Running   0          6h33m
cpu-loader-ab2d3s 1/1     Running   0          6h33m
cpu-loader-kip9a  1/1     Running   0          6h33m
foobar         1/1     Running   0          6h34m
front-end-6bc87b9748-24rcm 1/1     Running   0          5m4s
front-end-6bc87b9748-hd5wp 1/1     Running   0          5m2s
kucc1          0/2     ContainerCreating 0          3s
nginx-kusc00401 1/1     Running   0          2m28s
webserver-84c89df75-2dljn 1/1     Running   0          6h38m
webserver-84c89df75-8d8x2  1/1     Running   0          6h38m
webserver-84c89df75-z5zz4 1/1     Running   0          3m51s
student@node-1:~$
```

---

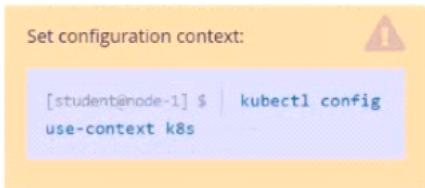
**Answer: A**

---

## Question: 2

---

Task Weight: 4%



Task

Scale the deployment webserver to 3 pods.

A . Explanation:

Solution:

```
student@node-1:~$ kubectl scale deploy webserver --replicas=3
deployment.apps/webserver scaled
student@node-1:~$ kubectl scale deploy webserver --replicas=3
```

---

**Answer: A**

---

## Question: 3

---

Score: 13%

Set configuration context: 

```
[student@node-1] $ | kube
ctl config use-context w
k8s
```

## Task

A Kubernetes worker node, named wk8s-node-0 is in state NotReady. Investigate why this is the case, and perform any appropriate steps to bring the node to a Ready state, ensuring that any changes are made permanent.

You can `ssh` to the failed node using: 

```
[student@node-1] $ | ssh
wk8s-node-0
```

You can assume elevated privileges on the node with the following command:

```
[student@wk8s-node-0] $ | sudo -i
```

A . Explanation:

Solution:

`sudo -i`

`systemctl status kubelet`

`systemctl start kubelet`

`systemctl enable kubelet`

---

**Answer: A**

---

## Question: 4

---

Score: 5%

Set configuration context: 

```
[student@node-1] $ | kube
ctl config use-context k
8s
```

## Task

From the pod label name=cpu-utilizer, find pods running high CPU workloads and write the name of the pod consuming most CPU to the file /opt/KUTR00401/KUTR00401.txt (which already exists).

A . Explanation:

Solution:

```
kubectl top -l name=cpu-user -A  
echo 'pod name' >> /opt/KUT00401/KUT00401.txt
```

---

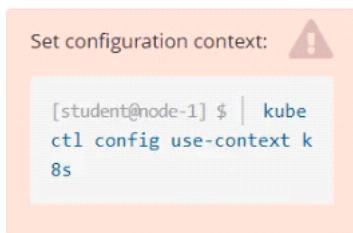
**Answer: A**

---

## **Question: 5**

---

Score:7%



Context

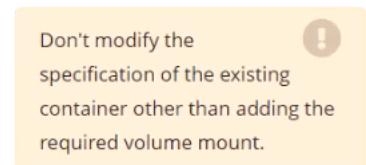
An existing Pod needs to be integrated into the Kubernetes built-in logging architecture (e. g. kubectl logs). Adding a streaming sidecar container is a good and common way to accomplish this requirement.

Task

Add a sidecar container named sidecar, using the busybox Image, to the existing Pod big-corp-app. The new sidecar container has to run the following command:

```
/bin/sh -c tail -n+1 -f /var/log/big-corp-app.log
```

Use a Volume, mounted at /var/log, to make the log file big-corp-app.log available to the sidecar container.



A . Explanation:

Solution:

```
#  
kubectl get pod big-corp-app -o yaml  
#  
apiVersion: v1  
kind: Pod  
metadata:
```

```

name: big-corp-app
spec:
  containers:
    - name: big-corp-app
      image: busybox
      args:
        - /bin/sh
        - -c
        - >
          i=0;
          while true;
          do
            echo '$(date) INFO $i' >> /var/log/big-corp-app.log;
            i=$((i+1));
            sleep 1;
          done
      volumeMounts:
        - name: logs
          mountPath: /var/log
    - name: count-log-1
      image: busybox
      args: [/bin/sh, -c, 'tail -n+1 -f /var/log/big-corp-app.log']
      volumeMounts:
        - name: logs
          mountPath: /var/log
      volumes:
        - name: logs
          emptyDir: {}
#
kubectl logs big-corp-app -c count-log-1

```

---

**Answer: A**

---

## **Question: 6**

---

Score: 5%

Set configuration context: 

```
[student@node-1] $ | kube
ctl config use-context k
8s
```

Task

Monitor the logs of pod bar and:

\* Extract log lines corresponding to error file-not-found

\* Write them to /opt/KUTR00101/bar

A . Explanation:

Solution:

```
kubectl logs bar | grep 'unable-to-access-website' > /opt/KUTR00101/bar  
cat /opt/KUTR00101/bar
```

---

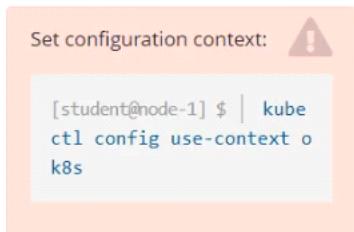
**Answer: A**

---

## **Question: 7**

---

Score:7%



Task

Create a new PersistentVolumeClaim

\* Name: pv-volume

\* Class: csi-hostpath-sc

\* Capacity: 10Mi

Create a new Pod which mounts the PersistentVolumeClaim as a volume:

\* Name: web-server

\* Image: nginx

\* Mount path: /usr/share/nginx/html

Configure the new Pod to have ReadWriteOnce access on the volume.

Finally, using kubectl edit or kubectl patch expand the PersistentVolumeClaim to a capacity of 70Mi and record that change.

A . Explanation:

Solution:

```
vi pvc.yaml
```

```
storageclass pvc
```

```
apiVersion: v1
```

```
kind: PersistentVolumeClaim
metadata:
name: pv-volume
spec:
accessModes:
- ReadWriteOnce
volumeMode: Filesystem
resources:
requests:
storage: 10Mi
storageClassName: csi-hostpath-sc
# vi pod-pvc.yaml
apiVersion: v1
kind: Pod
metadata:
name: web-server
spec:
containers:
- name: web-server
image: nginx
volumeMounts:
- mountPath: '/usr/share/nginx/html'
name: my-volume
volumes:
- name: my-volume
persistentVolumeClaim:
claimName: pv-volume
# craete
kubectl create -f pod-pvc.yaml
#edit
kubectl edit pvc pv-volume --record
```

---

**Answer: A**

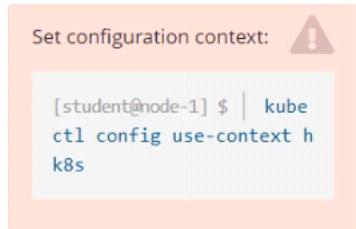
---

---

## Question: 8

---

Score: 4%



Task

Create a persistent volume with name app-data , of capacity 1Gi and access mode ReadOnlyMany. The type of volume is hostPath and its location is /srv/app-data .

A . Explanation:

Solution:

```
#vi pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-config
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadOnlyMany
  hostPath:
    path: /srv/app-config
#
kubectl create -f pv.yaml
```

---

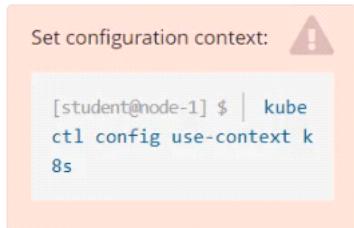
**Answer: A**

---

## **Question: 9**

---

Score: 4%



Task

Create a pod named kucc8 with a single app container for each of the following images running inside (there may be between 1 and 4 images specified): nginx + redis + memcached .

A . Explanation:

Solution:

```
kubectl run kucc8 --image=nginx --dry-run -o yaml > kucc8.yaml
# vi kucc8.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  name: kucc8
spec:
  containers:
```

```
- image: nginx
name: nginx
- image: redis
name: redis
- image: memcached
name: memcached
- image: consul
name: consul
#
kubectl create -f kucc8.yaml
#12.07
```

---

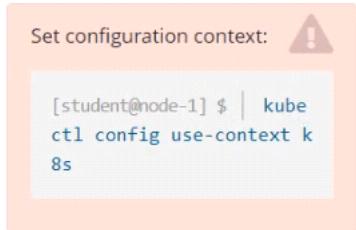
**Answer: A**

---

## **Question: 10**

---

Score: 4%



Task

Check to see how many nodes are ready (not including nodes tainted NoSchedule ) and write the number to /opt/KUSC00402/kusc00402.txt.

A . Explanation:

Solution:

```
kubectl describe nodes | grep ready|wc -l
kubectl describe nodes | grep -i taint | grep -i noschedule |wc -l
echo 3 > /opt/KUSC00402/kusc00402.txt
#
kubectl get node | grep -i ready |wc -l
# taintsnoSchedule
kubectl describe nodes | grep -i taints | grep -i noschedule |wc -l
#
echo 2 > /opt/KUSC00402/kusc00402.txt
```

---

**Answer: A**

---

## **Question: 11**

---

Score: 4%

Set configuration context: 

```
[student@node-1] $ | kube
ctl config use-context k
8s
```

## Task

Schedule a pod as follows:

- \* Name: nginx-kusc00401
- \* Image: nginx
- \* Node selector: disk=ssd

A . Explanation:

Solution:

```
#yaml
apiVersion: v1
kind: Pod
metadata:
name: nginx-kusc00401
spec:
containers:
- name: nginx
image: nginx
imagePullPolicy: IfNotPresent
nodeSelector:
disk: spinning
#
kubectl create -f node-select.yaml
```

---

**Answer: A**

---

---

## Question: 12

---

Score: 4%

Set configuration context: 

```
[student@node-1] $ | kube
ctl config use-context k
8s
```

## Task

Scale the deployment presentation to 6 pods.

A . Explanation:

Solution:

kubectl get deployment

kubectl scale deployment.apps/presentation --replicas=6

---

**Answer: A**

---

## **Question: 13**

---

Score: 7%

Set configuration context:



```
[student@node-1] $ | kube
ctl config use-context k
8s
```

Task

Create a new nginx Ingress resource as follows:

\* Name: ping

\* Namespace: ing-internal

\* Exposing service hi on path /hi using service port 5678

The availability of service [hi](#)

can be checked using the following command, which should return hi :

```
[student@node-1] $ | curl
-kL <INTERNAL_IP>/hi
```

A . Explanation:

Solution:

vi ingress.yaml

#

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ping

namespace: ing-internal

spec:

```
rules:  
- http:  
paths:  
- path: /hi  
pathType: Prefix  
backend:  
service:  
name: hi  
port:  
number: 5678  
#  
kubectl create -f ingress.yaml
```

---

**Answer: A**

---

## **Question: 14**

---

Score: 7%



### Task

Reconfigure the existing deployment front-end and add a port specification named http exposing port 80/tcp of the existing container nginx.

Create a new service named front-end-svc exposing the container port http.

Configure the new service to also expose the individual Pods via a NodePort on the nodes on which they are scheduled.

A . Explanation:

Solution:

```
kubectl get deploy front-end  
kubectl edit deploy front-end -o yaml  
#port specification named http  
#service.yaml  
apiVersion: v1  
kind: Service  
metadata:  
name: front-end-svc  
labels:  
app: nginx  
spec:
```

```
ports:
- port: 80
protocol: tcp
name: http
selector:
app: nginx
type: NodePort
# kubectl create -f service.yaml
# kubectl get svc
# port specification named http
kubectl expose deployment front-end --name=front-end-svc --port=80 --targetPort=80 --type=NodePort
```

---

---

### Answer: A

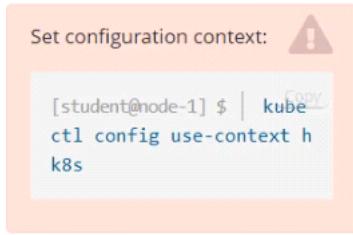
---

---

## Question: 15

---

Score: 7%



### Task

Create a new NetworkPolicy named allow-port-from-namespace in the existing namespace echo. Ensure that the new NetworkPolicy allows Pods in namespace my-app to connect to port 9000 of Pods in namespace echo.

Further ensure that the new NetworkPolicy:

- \* does not allow access to Pods, which don't listen on port 9000
- \* does not allow access from Pods, which are not in namespace my-app

A . Explanation:

Solution:

```
#network.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: allow-port-from-namespace
namespace: internal
spec:
podSelector:
matchLabels: {}
policyTypes:
```

```
- Ingress
ingress:
- from:
- podSelector: {
}
ports:
- protocol: TCP
port: 8080
#spec.podSelector namespace pod
kubectl create -f network.yaml
```

---

**Answer: A**

---

## **Question: 16**

---

Score: 7%

No configuration context  
change required for this  
task.

Ensure, however, that you have  
returned to the base node  
before starting to work on this  
task:

```
[student@mk8s-master-0] |  
$ |  
exit
```

## Task

Creating a snapshot of the given instance is expected to complete in seconds. If the operation seems to hang, something's likely wrong with your command. Use **CTRL + C** to cancel the operation and try again.

Next, restore an existing, previous snapshot located at /var/lib/backup/etcd-snapshot-previous.us.db

The following TLS certificates/key are supplied for connecting to the server with etcdctl :

- CA certificate:  
`/opt/KUIN00601/ca.crt`
- Client certificate:  
`/opt/KUIN00601/etcd-client.crt`
- Client key:  
`/opt/KUIN00601/etcd-client.key`

#### A . Explanation:

Solution:

#backup

```
ETCDCTL_API=3 etcdctl --endpoints='https://127.0.0.1:2379' --cacert=/opt/KUIN000601/ca.crt --cert=/opt/KUIN000601/etcd-client.crt --key=/opt/KUIN000601/etcd-client.key snapshot save /etc/data/etcd-snapshot.db
```

#restore

```
ETCDCTL_API=3 etcdctl --endpoints='https://127.0.0.1:2379' --cacert=/opt/KUIN000601/ca.crt --cert=/opt/KUIN000601/etcd-client.crt --key=/opt/KUIN000601/etcd-client.key snapshot restore /var/lib/backup/etcd-snapshot-prevoiys.db
```

---

**Answer: A**

---

### Question: 17

---

Score: 7%

Set configuration context: 

```
[student@node-1] $ | kube
ctl config use-context m
k8s
```

#### Task

Given an existing Kubernetes cluster running version 1.20.0, upgrade all of the Kubernetes control plane and node components on the master node only to version 1.20.1.

Be sure to drain the master node before upgrading it and uncordon it after the upgrade.

You can `ssh` to the master



node using:

```
[student@node-1] $ | ssh  
mk8s-master-0
```

You can assume elevated  
privileges on the master node  
with the following command:

```
[student@mk8s-master-0] |  
$  
sudo -i
```

You are also expected to upgrade kubelet and kubectl on the master node.

Do not upgrade the worker



nodes, etcd, the container  
manager, the CNI plugin, the  
DNS service or any other  
addons.

A . Explanation:

SOLUTION:

```
[student@node-1] > ssh ek8s  
kubectl cordon k8s-master  
kubectl drain k8s-master --delete-local-data --ignore-daemonsets --force  
apt-get install kubeadm=1.20.1-00 kubelet=1.20.1-00 kubectl=1.20.1-00 --disableexcludes=kubernetes  
kubeadm upgrade apply 1.20.1 --etcd-upgrade=false  
systemctl daemon-reload  
systemctl restart kubelet  
kubectl uncordon k8s-master
```

---

**Answer: A**

---

## Question: 18

---

Score: 4%

Set configuration context:



```
[student@node-1] $ | kube  
ctl config use-context e  
k8s
```

Task

Set the node named ek8s-node-1 as unavailable and reschedule all the pods running on it.

A . Explanation:

SOLUTION:

```
[student@node-1] > ssh ek8s
kubectl cordon ek8s-node-1
kubectl drain ek8s-node-1 --delete-local-data --ignore-daemonsets --force
```

---

**Answer: A**

---

## **Question: 19**

---

Score: 4%

Set configuration context: 

```
[student@node-1] $ | kube
ctl config use-context k
8s
```

Context

You have been asked to create a new ClusterRole for a deployment pipeline and bind it to a specific ServiceAccount scoped to a specific namespace.

Task

Create a new ClusterRole named deployment-clusterrole, which only allows to create the following resource types:

- \* Deployment
- \* StatefulSet
- \* DaemonSet

Create a new ServiceAccount named cicd-token in the existing namespace app-team1.

Bind the new ClusterRole deployment-clusterrole to the new ServiceAccount cicd-token , limited to the namespace app-team1.

A . Explanation:

Solution:

Task should be complete on node k8s -1 master, 2 worker for this connect use command

```
[student@node-1] > ssh k8s
kubectl create clusterrole deployment-clusterrole --verb=create --resource=deployments,statefulsets,daemonsets
kubectl create serviceaccount cicd-token --namespace=app-team1
kubectl create rolebinding deployment-clusterrole --clusterrole=deployment-clusterrole --
serviceaccount=default:cicd-token --namespace=app-team1
```

---

---

**Answer: A**

---

### **Question: 20**

List "nginx-dev" and "nginx-prod" pod and delete those pods

A . Explanation:

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

---

---

**Answer: A**

---

### **Question: 21**

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

A . Explanation:

```
kubectl get pods -o=jsonpath='{.items[*]['metadata.name',  
'metadata.namespace']}
```

---

---

**Answer: A**

---

### **Question: 22**

List all the pods sorted by created timestamp

A . Explanation:

```
kubect1 get pods--sort-by=.metadata.creationTimestamp
```

---

---

**Answer: A**

---

### **Question: 23**

List all the pods sorted by name

A . Explanation:

```
kubectl get pods --sort-by=.metadata.name
```

---

---

**Answer: A**

---

## **Question: 24**

---

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

A . Explanation:

```
kubectl get po -o=custom-columns='POD_NAME:.metadata.name,  
POD_STATUS:.status.containerStatuses[].state'
```

---

**Answer: A**

---

---

## **Question: 25**

---

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

A . Explanation:

```
// 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
```

---

**Answer: A**

---

---

## **Question: 26**

---

Create a busybox pod and add "sleep 3600" command

A . Explanation:

```
kubectl run busybox --image=busybox --restart=Never -- /bin/sh -c  
'sleep 3600'
```

---

**Answer: A**

---

---

## **Question: 27**

---

Check the Image version of nginx-dev pod using jsonpath

A . Explanation:

```
kubectl get po nginx-dev -o  
jsonpath='{.spec.containers[].image}{\'\n'}
```

---

**Answer: A**

---

---

## **Question: 28**

---

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

A . Explanation:

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

---

**Answer: A**

---

---

## **Question: 29**

---

Get IP address of the pod -- "nginx-dev"

A . Explanation:

```
Kubect1 get po -o wide
```

Using JsonPath

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

---

**Answer: A**

---

---

## **Question: 30**

---

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.

A . Explanation:

```
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
```

---

---

### Answer: A

---

---

## Question: 31

---

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

A . Explanation:

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
```

---

---

**Answer: A**

---

### **Question: 32**

List all the pods sorted by name

A . Explanation:

```
kubectl get pods --sort-by=.metadata.name
```

---

---

**Answer: A**

---

### **Question: 33**

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

A . Explanation:

```
kubectl get po -o=custom-columns='POD_NAME:.metadata.name,  
POD_STATUS:.status.containerStatuses[].state'
```

---

---

**Answer: A**

---

### **Question: 34**

Check the image version in pod without the describe command

A . Explanation:

```
kubectl get po nginx -o  
jsonpath='{.spec.containers[].image}\n'
```

---

---

**Answer: A**

---

### **Question: 35**

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

A . Explanation:

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

---

---

**Answer: A**

---

## **Question: 36**

---

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

A . Explanation:

```
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
```

---

**Answer: A**

---

---

## **Question: 37**

---

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

A . Explanation:

```
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'
```

---

**Answer: A**

---

---

## **Question: 38**

---

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

A . Explanation:

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

---

**Answer: A**

---

---

## **Question: 39**

---

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

A . Explanation:

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

---

**Answer: A**

---

---

## **Question: 40**

---

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

A . Explanation:

```
kubectl run nginx --image=nginx --restart=Never --port=80
```

---

**Answer: A**

---

---

## **Question: 41**

---

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

A . Explanation:

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

---

**Answer: A**

---

---

## **Question: 42**

---

Create a nginx pod with label env=test in engineering namespace

A . Explanation:

```
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
```

---

**Answer: A**

---

---

## **Question: 43**

---

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

A . Explanation:

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

---

**Answer: A**

---

---

## **Question: 44**

---

Create a persistent volume with name app-data, of capacity 2Gi and access mode ReadWriteMany. The type of volume is hostPath and its location is /srv/app-data.

A . Explanation:

solution

Persistent Volume

A persistent volume is a piece of storage in a Kubernetes cluster. PersistentVolumes are a cluster-level resource like nodes, which don't belong to any namespace. It is provisioned by the administrator and has a particular file size. This way, a developer deploying their app on Kubernetes need not know the underlying infrastructure. When the developer needs a certain amount of persistent storage for their application, the system administrator configures the cluster so that they consume the PersistentVolume provisioned in an easy way.

Creating Persistent Volume

```
kind: PersistentVolume
```

```
apiVersion: v1
```

```
metadata:
```

```
name:app-data
```

```
spec:
```

```
capacity: # defines the capacity of PV we are creating
```

```
storage: 2Gi #the amount of storage we are tying to claim
```

```
accessModes: # defines the rights of the volume we are creating
```

```
- ReadWriteMany
```

```
hostPath:
```

```
path: '/srv/app-data' # path to which we are creating the volume
```

Challenge

Create a Persistent Volume named app-data, with access mode ReadWriteMany, storage classname shared, 2Gi of storage capacity and the host path /srv/app-data.

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-data
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: /srv/app-data
  storageClassName: shared
~
```

“app-data.yaml” 12L, 194C

2. Save the file and create the persistent volume.

```
njerry191@cloudshell:~ (extreme-clone-265411)$ kubectl create -f pv.yaml
persistentvolume/pv created
```

3. View the persistent volume.

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS
app-data	2Gi	RWX	Retain	Available		shared

Our persistent volume status is available meaning it is available and it has not been mounted yet. This status will change when we mount the persistentVolume to a persistentVolumeClaim.

#### PersistentVolumeClaim

In a real ecosystem, a system admin will create the PersistentVolume then a developer will create a PersistentVolumeClaim which will be referenced in a pod. A PersistentVolumeClaim is created by specifying the minimum size and the access mode they require from the persistentVolume.

#### Challenge

Create a Persistent Volume Claim that requests the Persistent Volume we had created above. The claim should request 2Gi. Ensure that the Persistent Volume Claim has the same storageClassName as the persistentVolume you had previously created.

kind: PersistentVolume

apiVersion: v1

```

metadata:
name:app-data
spec:
accessModes:
- ReadWriteMany
resources:
requests:
storage: 2Gi
storageClassName: shared
2. Save and create the pvc
njerry191@cloudshell:~ (extreme-clone-2654111)$ kubectl create -f app-data.yaml
persistentvolumeclaim/app-data created
3. View the pvc

```

```

njerry191@cloudshell:~ (extreme-clone-265411) $ kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
pv        Bound     pv       512m       RWX           sha

```

4. Let's see what has changed in the pv we had initially created.

```

njerry191@cloudshell:~ (extreme-clone-265411) $ kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS    CLAIM           STORAGECLASS   REASON
pv        512m       RWX           Retain          Bound    default/pv     shared          16m

```

Our status has now changed from available to bound.

5. Create a new pod named myapp with image nginx that will be used to Mount the Persistent Volume Claim with the path /var/app/config.

Mounting a Claim

```

apiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
name: app-data
spec:
volumes:
- name: configpvc
persistenVolumeClaim:
claimName: app-data
containers:
- image: nginx
name: app
volumeMounts:
- mountPath: '/srv/app-data'
name: configpvc

```

---

**Answer: A**

---



---

**Question: 45**

---

Given a partially-functioning Kubernetes cluster, identify symptoms of failure on the cluster.

Determine the node, the failing service, and take actions to bring up the failed service and restore the health of the cluster. Ensure that any changes are made permanently.

You can ssh to the relevant nodes (bk8s-master-0 or bk8s-node-0) using:

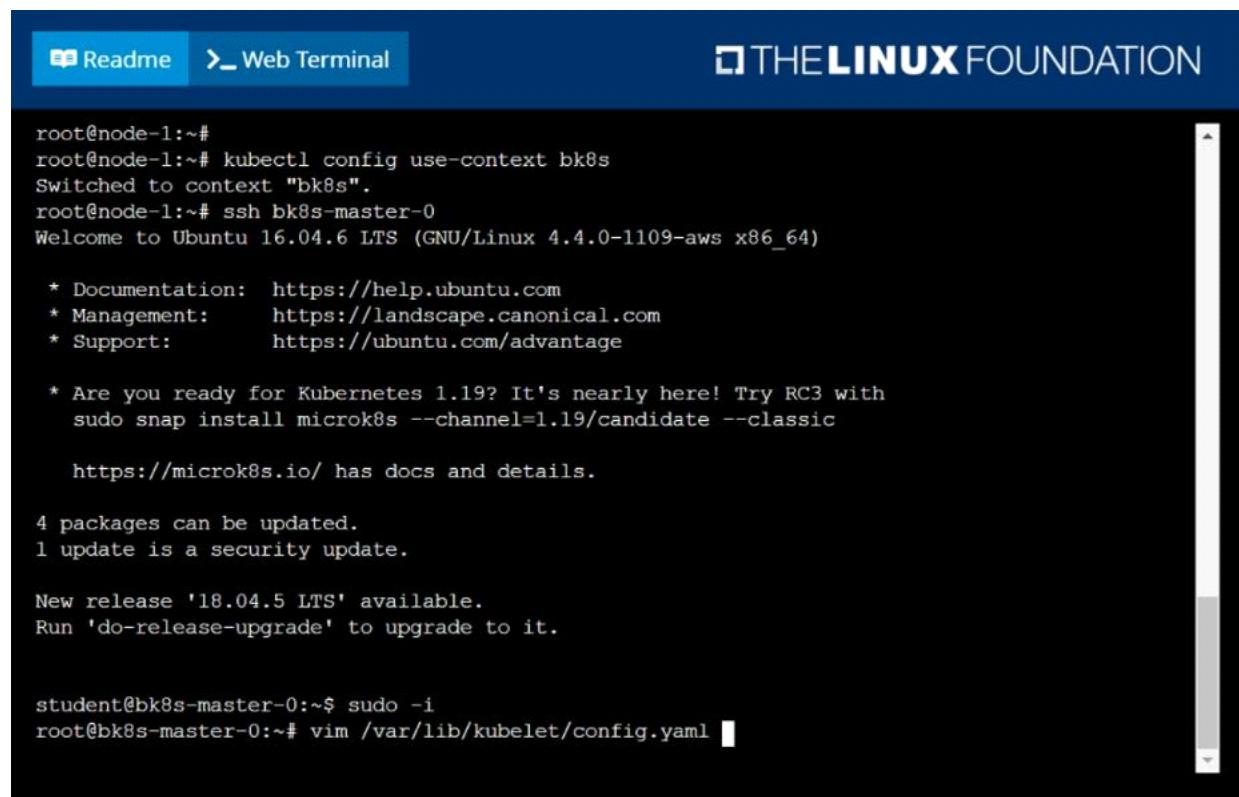
```
[student@node-1] $ ssh <nodename>
```

You can assume elevated privileges on any node in the cluster with the following command:

```
[student@nodename] $ | sudo --i
```

A . Explanation:

solution



The screenshot shows a web-based terminal interface with a dark theme. At the top, there are two buttons: "Readme" and "Web Terminal". To the right of the buttons, the text "THE LINUX FOUNDATION" is displayed. The main area is a terminal window showing the following command-line session:

```
root@node-1:~# kubectl config use-context bk8s
Switched to context "bk8s".
root@node-1:~# ssh bk8s-master-0
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1109-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Are you ready for Kubernetes 1.19? It's nearly here! Try RC3 with
   sudo snap install microk8s --channel=1.19/candidate --classic

 https://microk8s.io/ has docs and details.

4 packages can be updated.
1 update is a security update.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

student@bk8s-master-0:~$ sudo -i
root@bk8s-master-0:~# vim /var/lib/kubelet/config.yaml
```

 Readme

 Web Terminal

 THE LINUX FOUNDATION

```
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
rotateCertificates: true
runtimeRequestTimeout: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
:wq
```

 Readme

 Web Terminal

 THE LINUX FOUNDATION

<https://microk8s.io/> has docs and details.

4 packages can be updated.

1 update is a security update.

New release '18.04.5 LTS' available.

Run 'do-release-upgrade' to upgrade to it.

```
student@bk8s-master-0:~$ sudo -i
root@bk8s-master-0:~# vim /var/lib/kubelet/config.yaml
root@bk8s-master-0:~# systemctl restart kubelet
root@bk8s-master-0:~# systemctl enable kubelet
root@bk8s-master-0:~# kubectl get nodes
```

```
NAME      STATUS   ROLES   AGE     VERSION
bk8s-master-0  Ready    master  77d    v1.18.2
bk8s-node-0   Ready    <none>  77d    v1.18.2
root@bk8s-master-0:~#
root@bk8s-master-0:~# exit
logout
student@bk8s-master-0:~$ exit
logout
Connection to 10.250.4.77 closed.
root@node-1:~#
```

---

Answer: A

---

---

Question: 46

---

For this item, you will have to ssh to the nodes ik8s-master-0 and ik8s-node-0 and complete all tasks on these nodes. Ensure that you return to the base node (hostname: node-1) when you have completed this item.

## Context

As an administrator of a small development team, you have been asked to set up a Kubernetes cluster to test the viability of a new application.

## Task

You must use kubeadm to perform this task. Any kubeadm invocations will require the use of the --ignore-preflight-errors=all option.

Configure the node ik8s-master-0 as a master node. .

Join the node ik8s-node-0 to the cluster.

### A . Explanation:

#### solution

You must use the kubeadm configuration file located at /etc/kubeadm.conf when initializing your cluster.

You may use any CNI plugin to complete this task, but if you don't have your favourite CNI plugin's manifest URL at hand, Calico is one popular option: <https://docs.projectcalico.org/v3.14/manifests/calico.yaml>

Docker is already installed on both nodes and apt has been configured so that you can install the required tools.

---

**Answer: A**

---

## **Question: 47**

---

Configure the kubelet systemd- managed service, on the node labelled with name=wk8s-node-1, to launch a pod containing a single container of Image httpd named webtool automatically. Any spec files required should be placed in the /etc/kubernetes/manifests directory on the node.

You can ssh to the appropriate node using:

```
[student@node-1] $ ssh wk8s-node-1
```

You can assume elevated privileges on the node with the following command:

```
[student@wk8s-node-1] $ | sudo --i
```

### A . Explanation:

#### solution

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

```
root@node-1:~#
root@node-1:~# kubectl config use-context wk8s
Switched to context "wk8s".
root@node-1:~# ssh wk8s-node-1
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1109-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Are you ready for Kubernetes 1.19? It's nearly here! Try RC3 with
   sudo snap install microk8s --channel=1.19/candidate --classic

 https://microk8s.io/ has docs and details.

4 packages can be updated.
1 update is a security update.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

student@wk8s-node-1:~$ sudo -i
root@wk8s-node-1:~# vim /var/lib/kubelet/config.yaml
```

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

```
clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
rotateCertificates: true
runtimeRequestTimeout: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
:wq
```

[Readme](#) >\_ [Web Terminal](#)

THE LINUX FOUNDATION

```
root@node-1:~# ssh wk8s-node-1
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1109-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Are you ready for Kubernetes 1.19? It's nearly here! Try RC3 with
   sudo snap install microk8s --channel=1.19/candidate --classic

   https://microk8s.io/ has docs and details.

4 packages can be updated.
1 update is a security update.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

```
student@wk8s-node-1:~$ sudo -i  
root@wk8s-node-1:~# vim /var/lib/kubelet/config.yaml  
root@wk8s-node-1:~# cd /etc/kubernetes/manifests  
root@wk8s-node-1:/etc/kubernetes/manifests#  
root@wk8s-node-1:/etc/kubernetes/manifests# vim pod.yaml
```

[Readme](#)

THE **LINUX** FOUNDATION

```
apiVersion: v1
kind: Pod
metadata:
  name: webtool
spec:
  containers:
    - name: webtool
      image: httpd
```

```
https://microk8s.io/ has docs and details.

4 packages can be updated.
1 update is a security update.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

student@wk8s-node-1:~$ sudo -i
root@wk8s-node-1:~# vim /var/lib/kubelet/config.yaml
root@wk8s-node-1:~# cd /etc/kubernetes/manifests
root@wk8s-node-1:/etc/kubernetes/manifests#
root@wk8s-node-1:/etc/kubernetes/manifests# vim pod.yaml
root@wk8s-node-1:/etc/kubernetes/manifests# systemctl restart kubelet
root@wk8s-node-1:/etc/kubernetes/manifests# systemctl enable kubelet
root@wk8s-node-1:/etc/kubernetes/manifests# exit
logout
student@wk8s-node-1:~$ exit
logout
Connection to 10.250.5.39 closed.
root@node-1:~# k get po
NAME           READY   STATUS    RESTARTS   AGE
webtool-wk8s-node-1   1/1     Running   0          11s
root@node-1:~#
```

---

**Answer: A**

---

## Question: 48

---

A Kubernetes worker node, named wk8s-node-0 is in state NotReady. Investigate why this is the case, and perform any appropriate steps to bring the node to a Ready state, ensuring that any changes are made permanent.

You can ssh to the failed node using:

```
[student@node-1] $ | ssh Wk8s-node-0
```

You can assume elevated privileges on the node with the following command:

```
[student@w8ks-node-0] $ | sudo --
```

A . Explanation:  
solution

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

```
root@node-1:~# kubectl config use-context wk8s
Switched to context "wk8s".
root@node-1:~# k get nodes
NAME        STATUS   ROLES   AGE    VERSION
wk8s-master-0  Ready    master  77d    v1.18.2
wk8s-node-0    NotReady <none>  77d    v1.18.2
wk8s-node-1    Ready    <none>  77d    v1.18.2
root@node-1:~# ssh wk8s-node-0
```

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

```
wk8s-node-0    NotReady  <none>  77d    v1.18.2
wk8s-node-1    Ready     <none>  77d    v1.18.2
root@node-1:~# ssh wk8s-node-0
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1109-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Are you ready for Kubernetes 1.19? It's nearly here! Try RC3 with
   sudo snap install microk8s --channel=1.19/candidate --classic

   https://microk8s.io/ has docs and details.

4 packages can be updated.
1 update is a security update.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

student@wk8s-node-0:~$ sudo -i
root@wk8s-node-0:~# systemctl restart kubelet
root@wk8s-node-0:~# systemctl enable kubelet
```

```
https://microk8s.io/ has docs and details.

4 packages can be updated.
1 update is a security update.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

student@wk8s-node-0:~$ sudo -i
root@wk8s-node-0:~# systemctl restart kubelet
root@wk8s-node-0:~# systemctl enable kubelet
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /lib/sys
temd/system/kubelet.service.
root@wk8s-node-0:~# exit
logout
student@wk8s-node-0:~$ exit
logout
Connection to 10.250.5.34 closed.
root@node-1:~# k get nodes
NAME        STATUS   ROLES     AGE    VERSION
wk8s-master-0  Ready    master   77d    v1.18.2
wk8s-node-0    Ready    <none>   77d    v1.18.2
wk8s-node-1    Ready    <none>   77d    v1.18.2
root@node-1:~# 
```

---

**Answer: A**

---

## Question: 49

---

Set the node named ek8s-node-1 as unavailable and reschedule all the pods running on it.

A . Explanation:  
solution

```
root@node-1:~# kubectl config use-context ek8s
Switched to context "ek8s".
root@node-1:~# k drain ek8s-node-1 --ignore-daemonsets --delete-local-data --force
node/ek8s-node-1 cordoned
WARNING: ignoring DaemonSet-managed Pods: kube-system/kube-flannel-ds-amd64-qj7w8, kube-syst
em/kube-proxy-x7xkv
evicting pod default/nginx-568f5649b8-c9zkj
evicting pod kube-system/metrics-server-64b57fd654-cktk5
[]
```

---

**Answer: A**

---

---

**Question: 50**

---

Create a snapshot of the etcd instance running at <https://127.0.0.1:2379>, saving the snapshot to the file path /srv/data/etcd-snapshot.db.

The following TLS certificates/key are supplied for connecting to the server with etcdctl:

CA certificate: /opt/KUCM00302/ca.crt

Client certificate: /opt/KUCM00302/etcd-client.crt

Client key: /opt/KUCM00302/etcd-client.key

A . Explanation:

solution

```
root@node-1:~# ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/opt/KUCM00302/ca.crt --cert=/opt/KUCM00302/etcd-client.crt --key=/opt/KUCM00302/etcd-client.key snapshot save /srv/data/etcd-snapshot.db
{"level":"info","ts":1598530470.8313155,"caller":"snapshot/v3_snapshot.go:110","msg":"create temporary db file","path":"/srv/data/etcd-snapshot.db.part"}
{"level":"warn","ts":2020-08-27T12:14:30.838Z,"caller":"clientv3/retry_interceptor.go:116","msg":"retry stream intercept"}
{"level":"info","ts":1598530470.8388612,"caller":"snapshot/v3_snapshot.go:121","msg":"fetching snapshot","endpoint":"https://127.0.0.1:2379"}
{"level":"info","ts":1598530470.8570414,"caller":"snapshot/v3_snapshot.go:134","msg":"fetched snapshot","endpoint":"https://127.0.0.1:2379","took":0.025676157}
{"level":"info","ts":1598530470.8571067,"caller":"snapshot/v3_snapshot.go:143","msg":"saved","path":"/srv/data/etcd-snapshot.db"}
Snapshot saved at /srv/data/etcd-snapshot.db
root@node-1:~#
```

---

**Answer: A**

---

---

**Question: 51**

---

Create a deployment as follows:

Name: nginx-random

Exposed via a service nginx-random

Ensure that the service & pod are accessible via their respective DNS records

The container(s) within any pod(s) running as a part of this deployment should use the nginx Image

Next, use the utility nslookup to look up the DNS records of the service & pod and write the output to /opt/KUNW00601/service.dns and /opt/KUNW00601/pod.dns respectively.

A . Explanation:

Solution:

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

```
root@node-1:~#  
root@node-1:~# k create deploy nginx-random --image=nginx  
deployment.apps/nginx-random created  
root@node-1:~# k expose deploy nginx-random --name=nginx-random --port=80 --target-port=80  
service/nginx-random exposed  
root@node-1:~# vim dns.yaml
```

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

```
apiVersion: v1  
kind: Pod  
metadata:  
  name: busybox1  
  labels:  
    name: busybox  
spec:  
  containers:  
  - image: busybox:1.28  
    command:  
      - sleep  
      - "3600"  
    name: busybox
```

```
root@node-1:~# k create deploy nginx-random --image=nginx
deployment.apps/nginx-random created
root@node-1:~# k expose deploy nginx-random --name=nginx-random --port=80 --target-port=80
service/nginx-random exposed
root@node-1:~# vim dns.yaml
root@node-1:~# k create -f dns.yaml
pod/busybox1 created
root@node-1:~# k get po -o wide | grep nginx-random
nginx-random-6d5766bbdc-ptzv2 1/1     Running   0          103s   10.244.2.16   k8s-node-
1 <none>           <none>
root@node-1:~# k exec -it busybox1 -- nslookup nginx-random
Server:  10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      nginx-random
Address 1: 10.111.37.132 nginx-random.default.svc.cluster.local
root@node-1:~# k exec -it busybox1 -- nslookup nginx-random > /opt/KUNW00601/service.dns
root@node-1:~# k exec -it busybox1 -- nslookup 10-244-2-16.default.pod
Server:  10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      10-244-2-16.default.pod
Address 1: 10.244.2.16 10-244-2-16.nginx-random.default.svc.cluster.local
root@node-1:~# k exec -it busybox1 -- nslookup 10-244-2-16.default.pod > /opt/KUNW00601/pod
.dns
```

---

**Answer: A**

---

---

## Question: 52

---

From the pod label name=cpu-utilizer, find pods running high CPU workloads and write the name of the pod consuming most CPU to the file /opt/KUTR00102/KUTR00102.txt (which already exists).

A . Explanation:  
solution

[Readme](#) [Web Terminal](#)

THE LINUX FOUNDATION

```
root@node-1:~# k top po -l name(cpu-utilizer)
NAME          CPU(cores)   MEMORY(bytes)
cpu-utilizer-98b9se  60m        7Mi
cpu-utilizer-ab2d3s  14m        7Mi
cpu-utilizer-kipb9a  45m        7Mi
root@node-1:~# vim /opt/KUTR00102/KUTR00102.txt
```

 Readme  Web Terminal

THE LINUX FOUNDATION

## Answer: A

## Question: 53

Check to see how many worker nodes are ready (not including nodes tainted NoSchedule) and write the number to /opt/KUCC00104/kucc00104.txt.

#### A . Explanation:

## solution

```
Readme >_ Web Terminal THE LINUX FOUNDATION

root@node-1:~# k scale deploy webserver --replicas=6
deployment.apps/webserver scaled
root@node-1:~# k get deploy
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
nginx-app   3/3     3           3           29m
webserver   6/6     6           6           6h50m
root@node-1:~#
root@node-1:~# k get nodes
NAME        STATUS   ROLES   AGE   VERSION
k8s-master-0 Ready    master   77d   v1.18.2
k8s-node-0  Ready    <none>  77d   v1.18.2
k8s-node-1  Ready    <none>  77d   v1.18.2
root@node-1:~# vim /opt/KUCC00104/kucc00104.txt
```

---

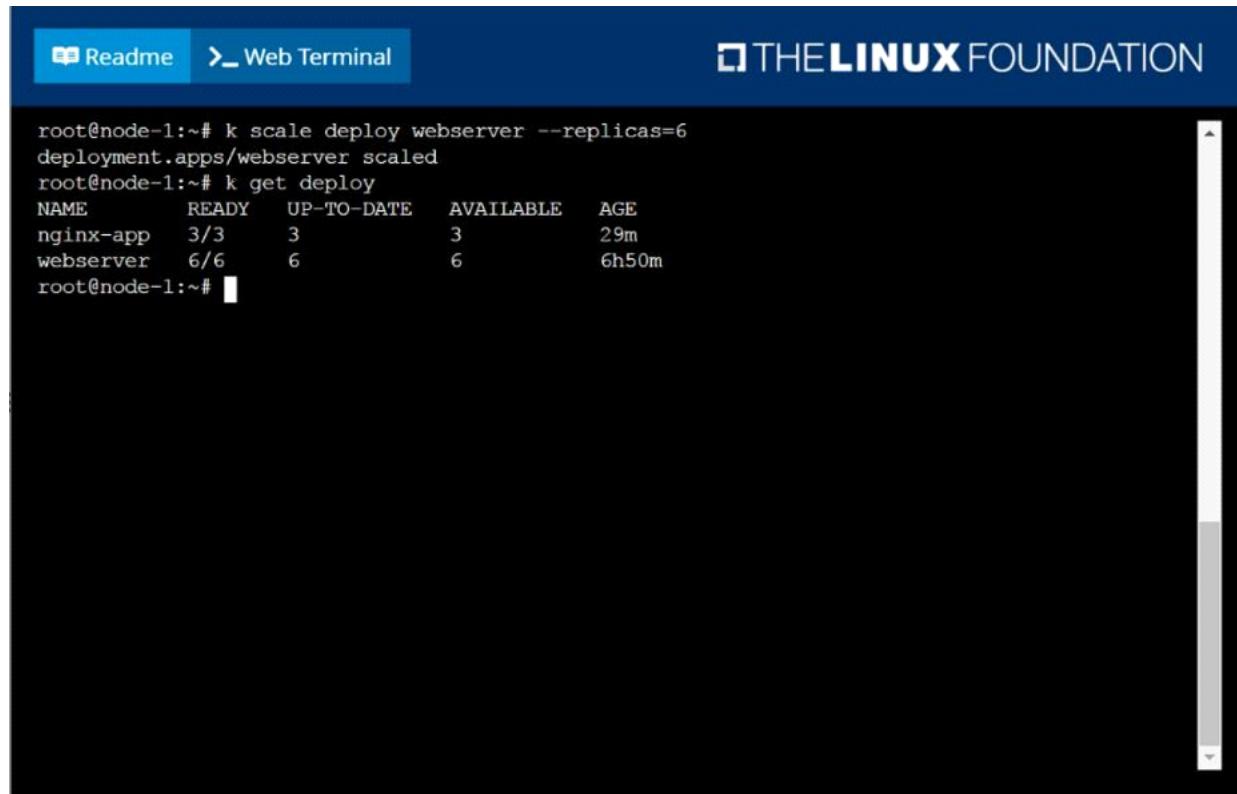
**Question: 54**

---

Scale the deployment webserver to 6 pods.

A . Explanation:

solution



The screenshot shows a web-based terminal interface with a blue header bar. On the left, there are two buttons: 'Readme' and 'Web Terminal'. On the right, the 'THE LINUX FOUNDATION' logo is visible. The main area is a black terminal window displaying the following command and its output:

```
root@node-1:~# k scale deploy webserver --replicas=6
deployment.apps/webserver scaled
root@node-1:~# k get deploy
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx-app  3/3     3           3           29m
webserver  6/6     6           6           6h50m
root@node-1:~#
```

---

**Question: 55**

---

Create a pod as follows:

Name: non-persistent-redis

container Image: redis

Volume with name: cache-control

Mount path: /data/redis

The pod should launch in the staging namespace and the volume must not be persistent.

## A . Explanation: solution

Readme > Web Terminal

THE LINUX FOUNDATION

```
root@node-1:~#  
root@node-1:~#  
root@node-1:~# vim volume.yaml
```

```
root@node-1:~#  
root@node-1:~#  
root@node-1:~# vim volume.yaml  
root@node-1:~# k create -f volume.yaml  
pod/non-persistent-redis created  
root@node-1:~# k get po -n staging  
NAME           READY   STATUS    RESTARTS   AGE  
non-persistent-redis  1/1     Running   0          6s  
root@node-1:~#
```

---

**Answer: A**

---

---

**Question: 56**

---

Create a Kubernetes secret as follows:

Name: super-secret

password: bob

Create a pod named pod-secrets-via-file, using the redis Image, which mounts a secret named super-secret at /secrets.

Create a second pod named pod-secrets-via-env, using the redis Image, which exports password as CONFIDENTIAL

A . Explanation:  
solution

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

```
root@node-1:~# k create secret generic super-secret --from-literal=password=bob
secret/super-secret created
root@node-1:~# vim secret.yaml
```

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-secrets-via-file
spec:
  containers:
  - name: redis
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/secrets"
  volumes:
  - name: foo
    secret:
      secretName: super-secret
~
~
~
~
~
~
~
: w
```

```
root@node-1:~# k create -f secret.yaml
pod/pod-secrets-via-file created
root@node-1:~# vim secret1.yaml
root@node-1:~# k create -f secret1.yaml
pod/pod-secrets-via-env created
root@node-1:~# k get po
NAME          READY   STATUS    RESTARTS   AGE
cpu-utilizer-98b9se  1/1     Running   0          6h25m
cpu-utilizer-ab2d3s  1/1     Running   0          6h25m
cpu-utilizer-kipb9a  1/1     Running   0          6h25m
ds-kusc00201-2r2k9  1/1     Running   0          40m
ds-kusc00201-hzm9q  1/1     Running   0          40m
foo             1/1     Running   0          6h28m
front-end       1/1     Running   0          6h27m
hungry-bear      1/1     Running   0          36m
kucc8           3/3     Running   0          34m
nginx-app-848cfccf495-9prjh  1/1     Running   0          19m
nginx-app-848cfccf495-gl2kh  1/1     Running   0          19m
nginx-app-848cfccf495-pg2c8  1/1     Running   0          19m
nginx-kusc00101  1/1     Running   0          26m
pod-secrets-via-env  1/1     Running   0          4s
pod-secrets-via-file   1/1     Running   0          106s
webserver-84c55967f4-qzjcv  1/1     Running   0          6h43m
webserver-84c55967f4-t4791  1/1     Running   0          6h43m
root@node-1:~# []
```

---

**Answer: A**

---

---

## Question: 57

---

Create a file:

/opt/KUCC00302/kucc00302.txt that lists all pods that implement service baz in namespace development.

The format of the file should be one pod name per line.

A . Explanation:

solution

[Readme](#) > [Web Terminal](#)

THE LINUX FOUNDATION

```
root@node-1:~# k describe svc baz -n development
Name:          baz
Namespace:     development
Labels:        <none>
Annotations:   <none>
Selector:      name=foo
Type:          ClusterIP
IP:            10.104.252.175
Port:          <unset>  80/TCP
TargetPort:    9376/TCP
Endpoints:    10.244.1.5:9376,10.244.2.3:9376,10.244.2.6:9376
Session Affinity: None
Events:        <none>
root@node-1:~# k get po -l name=foo -n development
NAME           READY   STATUS    RESTARTS   AGE
pod-kucc00302-847878  1/1     Running   0          6h35m
pod-kucc00302-983457  1/1     Running   0          6h35m
pod-kucc00302-985953  1/1     Running   0          6h35m
root@node-1:~# k get po -l name=foo -n development -o NAME
pod/pod-kucc00302-847878
pod/pod-kucc00302-983457
pod/pod-kucc00302-985953
root@node-1:~# k get po -l name=foo -n development -o NAME > /opt/KUCC00302/kucc00302.txt
root@node-1:~# vim /opt/KUCC00302/kucc00302.txt
```

[Readme](#) [Web Terminal](#)

THE LINUX FOUNDATION

pod-kucc00302-847878  
pod-kucc00302-983457  
pod-kucc00302-965953

```
Name: baz
Namespace: development
Labels: <none>
Annotations: <none>
Selector: name=foo
Type: ClusterIP
IP: 10.104.252.175
Port: <unset> 80/TCP
TargetPort: 9376/TCP
Endpoints: 10.244.1.5:9376,10.244.2.3:9376,10.244.2.6:9376
Session Affinity: None
Events: <none>
root@node-1:~# k get po -l name=foo -n development
NAME READY STATUS RESTARTS AGE
pod-kucc00302-847878 1/1 Running 0 6h35m
pod-kucc00302-983457 1/1 Running 0 6h35m
pod-kucc00302-985953 1/1 Running 0 6h35m
root@node-1:~# k get po -l name=foo -n development -o NAME
pod/pod-kucc00302-847878
pod/pod-kucc00302-983457
pod/pod-kucc00302-985953
root@node-1:~# k get po -l name=foo -n development -o NAME > /opt/KUCC00302/kucc00302.txt
root@node-1:~# vim /opt/KUCC00302/kucc00302.txt
root@node-1:~# vim /opt/KUCC00302/kucc00302.txt
root@node-1:~#
```

---

**Answer: A**

---

---

## Question: 58

---

Create a deployment spec file that will:

Launch 7 replicas of the nginx Image with the label  
app\_runtime\_stage=dev

deployment name: kual00201

Save a copy of this spec file to /opt/KUAL00201/spec\_deployment.yaml

(or /opt/KUAL00201/spec\_deployment.json).

When you are done, clean up (delete) any new Kubernetes API object that you produced during this task.

A . Explanation:  
solution

 Readme  Web Terminal

 THE LINUX FOUNDATION

```
root@node-1:~# k create deploy kual00201 --image=nginx --dry-run=client -o yaml > /opt/KUAL00201/spec_deployment.yaml
root@node-1:~# vim /opt/KUAL00201/spec_deployment.yaml
```

 Readme  Web Terminal

 THE LINUX FOUNDATION

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app_runtime_stage: dev
  name: kual00201
spec:
  replicas: 7
  selector:
    matchLabels:
      app_runtime_stage: dev
  template:
    metadata:
      labels:
        app_runtime_stage: dev
    spec:
      containers:
      - image: nginx
        name: nginx
~
```

"/opt/KUAL00201/spec\_deployment.yaml" 19L, 320C written

---

Answer: A

---

Question: 59

Create a pod as follows:

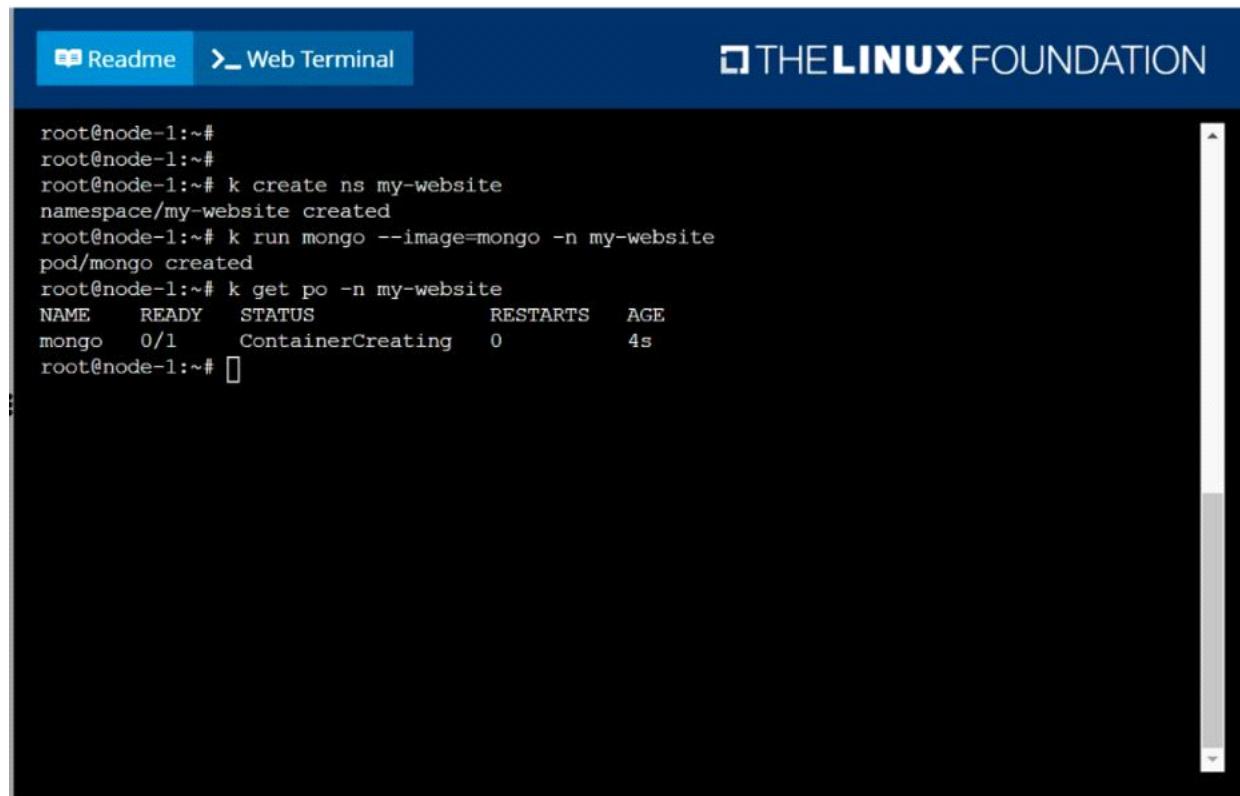
Name: mongo

Using Image: mongo

In a new Kubernetes namespace named: my-website

A . Explanation:

solution



```
root@node-1:~# k create ns my-website
namespace/my-website created
root@node-1:~# k run mongo --image=mongo -n my-website
pod/mongo created
root@node-1:~# k get po -n my-website
NAME      READY   STATUS        RESTARTS   AGE
mongo    0/1     ContainerCreating   0          4s
root@node-1:~#
```

---

**Answer: A**

---

## **Question: 60**

---

Create and configure the service front-end-service so it's accessible through NodePort and routes to the existing pod named front-end.

A . Explanation:

solution

```
root@node-1:~# k expose po
error: resource(s) were provided, but no name, label selector, or --all flag specified
See 'kubectl expose -h' for help and examples
root@node-1:~# k expose po  fron-end --name=front-end-service --port=80 --target-port=80 --t
ype=NodePort
Error from server (NotFound): pods "fron-end" not found
root@node-1:~# k expose po  front-end --name=front-end-service --port=80 --target-port=80 --t
ype=NodePort
service/front-end-service exposed
root@node-1:~# k get svc
NAME           TYPE      CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE
front-end-service  NodePort  10.103.221.227  <none>        80:31828/TCP  3s
kubernetes      ClusterIP  10.96.0.1       <none>        443/TCP       77d
root@node-1:~#
```

---

**Answer: A**

---

## **Question: 61**

---

Create a deployment as follows:

Name: nginx-app

Using container nginx with version 1.11.10-alpine

The deployment should contain 3 replicas

Next, deploy the application with new version 1.11.13-alpine, by performing a rolling update.

Finally, rollback that update to the previous version 1.11.10-alpine.

A . Explanation:

solution

[Readme](#) [Web Terminal](#)

THE LINUX FOUNDATION

```
root@node-1:~# k create deploy nginx-app --image=nginx:1.11.10-alpine --dry-run=client -o yaml > app.yaml  
root@node-1:~# vim app.yaml
```

 [Readme](#)

THE LINUX FOUNDATION

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-app
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
        - image: nginx:1.11.10-alpine
          name: nginx
~
```

```
root@node-1:~# k create deploy nginx-app --image=nginx:1.11.10-alpine --dry-run=client -o yaml > app.yaml
root@node-1:~# vim app.yaml
root@node-1:~# k create -f app.yaml
deployment.apps/nginx-app created
root@node-1:#
root@node-1:#
root@node-1:~# k set image deploy nginx-app nginx=nginx:1.11.13-alpine --record
deployment.apps/nginx-app image updated
root@node-1:~# k rollout undo deploy nginx-app
deployment.apps/nginx-app rolled back
root@node-1:~#
```

---

**Answer: A**

---

---

**Question: 62**

---

Schedule a pod as follows:

Name: nginx-kusc00101

Image: nginx

Node selector: disk=ssd

A . Explanation:  
solution

[Readme](#) [Web Terminal](#)

THE LINUX FOUNDATION

```
root@node-1:~# vim disk.yaml
```

 Readme >\_ Web Terminal

THE **LINUX** FOUNDATION

```
root@node-1:~# vim disk.yaml
root@node-1:~# k create -f disk.yaml
pod/nginx-kusc00101 created
root@node-1:~# k get po
NAME          READY   STATUS    RESTARTS   AGE
cpu-utilizer-98b9se  1/1     Running   0          5h59m
cpu-utilizer-ab2d3s  1/1     Running   0          5h59m
cpu-utilizer-kipb9a  1/1     Running   0          5h59m
ds-kusc00201-2r2k9  1/1     Running   0          13m
ds-kusc00201-hzm9q  1/1     Running   0          13m
foo            1/1     Running   0          6h1m
front-end       1/1     Running   0          6h1m
hungry-bear      1/1     Running   0          9m37s
kucc8           3/3     Running   0          7m37s
nginx-kusc00101  1/1     Running   0          9s
webserver-84c55967f4-qzjcv  1/1     Running   0          6h16m
webserver-84c55967f4-t4791  1/1     Running   0          6h16m
root@node-1:~#
```

---

**Answer: A**

---

### **Question: 63**

---

Create a pod named kucc8 with a single app container for each of the following images running inside (there may be between 1 and 4 images specified):

nginx + redis + memcached.

A . Explanation:  
solution

[Readme](#) [Web Terminal](#)

THE LINUX FOUNDATION

```

root@node-1:~# vim ds.yaml
iroot@node-1:~# k create -f ds.yaml
daemonset.apps/ds-kusc00201 created
root@node-1:~# k get ds
NAME      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
ds-kusc00201   2         2         2         2             2           <none>    4s

root@node-1:~# vim /opt/KUCC00108/pod-spec-KUCC00108.yaml
root@node-1:~# k create -f /opt/KUCC00108/pod-spec-KUCC00108.yaml
pod/hungry-bear created
root@node-1:~# k get po
NAME          READY   STATUS   RESTARTS   AGE
cpu-utilizer-98b9se   1/1     Running   0          5h50m
cpu-utilizer-ab2d3s   1/1     Running   0          5h50m
cpu-utilizer-kipb9a   1/1     Running   0          5h50m
ds-kusc00201-2r2k9   1/1     Running   0          4m50s
ds-kusc00201-hzm9q   1/1     Running   0          4m50s
foo            1/1     Running   0          5h52m
front-end       1/1     Running   0          5h52m
hungry-bear      1/1     Running   0          42s
webserver-84c55967f4-qzjcv  1/1     Running   0          6h7m
webserver-84c55967f4-t4791  1/1     Running   0          6h7m
root@node-1:~# k run nginx --image=nginx --dry-run=client -o yaml > nginx.yaml
root@node-1:~# vim nginx.yaml

```

 [Readme](#)

THE LINUX FOUNDATION

```
apiVersion: v1
kind: Pod
metadata:
  name: kucc8
spec:
  containers:
    - image: nginx
      name: nginx
    - image: redis
      name: redis
    - image: memcached
      name: memcached
~
```

```
cpu-utilizer-98b9se      1/1   Running     0      5h51m
cpu-utilizer-ab2d3s      1/1   Running     0      5h51m
cpu-utilizer-kipb9a      1/1   Running     0      5h51m
ds-kusc00201-2r2k9       1/1   Running     0      6m12s
ds-kusc00201-hzm9q       1/1   Running     0      6m12s
foo                      1/1   Running     0      5h54m
front-end                1/1   Running     0      5h53m
hungry-bear               1/1   Running     0      2m4s
kucc8                    0/3   ContainerCreating 0      4s
webserver-84c55967f4-qzjcv 1/1   Running     0      6h9m
webserver-84c55967f4-t4791 1/1   Running     0      6h9m
root@node-1:~# k get po
NAME                  READY   STATUS    RESTARTS   AGE
cpu-utilizer-98b9se    1/1    Running   0          5h52m
cpu-utilizer-ab2d3s    1/1    Running   0          5h52m
cpu-utilizer-kipb9a    1/1    Running   0          5h52m
ds-kusc00201-2r2k9     1/1    Running   0          6m31s
ds-kusc00201-hzm9q     1/1    Running   0          6m31s
foo                     1/1    Running   0          5h54m
front-end               1/1    Running   0          5h54m
hungry-bear              1/1    Running   0          2m23s
kucc8                  3/3    Running   0          23s
webserver-84c55967f4-qzjcv 1/1    Running   0          6h9m
webserver-84c55967f4-t4791 1/1    Running   0          6h9m
root@node-1:~#
```

---

**Answer: A**

---

## Question: 64

---

Perform the following tasks:

Add an init container to hungry-bear (which has been defined in spec file /opt/KUCC00108/pod-spec-KUC  
C00108.yaml  
)

The init container should create an empty file named  
/workdir/calm.txt

If /workdir/calm.txt is not detected, the pod should exit

Once the spec file has been updated with the init container definition, the pod should be created

A . Explanation:  
solution

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

```
root@node-1:~# vim ds.yaml
iroot@node-1:~# k create -f ds.yaml
daemonset.apps/ds-kusc00201 created
root@node-1:~# k get ds
NAME      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
ds-kusc00201   2         2         2         2           2           <none>     4s
root@node-1:~# vim /opt/KUCC00108/pod-spec-KUCC00108.yaml
```

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

```
apiVersion: v1
kind: Pod
metadata:
  name: hungry-bear
spec:
  volumes:
    - name: workdir
      emptyDir:
  containers:
    - name: checker
      image: alpine
      command: ["/bin/sh", "-c", "if [ -f /workdir/calm.txt ];
          then sleep 100000; else exit 1; fi"]
      volumeMounts:
        - name: workdir
          mountPath: /workdir
  initContainers:
    - name: create
      image: alpine
      command: ["/bin/sh", "-c", "touch /workdir/calm.txt"]
      volumeMounts:
        - name: workdir
          mountPath: /workdir
:wa
```

```
root@node-1:~# vim ds.yaml
iroot@node-1:~# k create -f ds.yaml
daemonset.apps/ds-kusc00201 created
root@node-1:~# k get ds
NAME      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
ds-kusc00201  2         2         2         2           2           <none>    4s
root@node-1:~# vim /opt/KUCC00108/pod-spec-KUCC00108.yaml
root@node-1:~# k create -f /opt/KUCC00108/pod-spec-KUCC00108.yaml
pod/hungry-bear created
root@node-1:~#
```

---

**Answer: A**

---

---

**Question: 65**

---

Ensure a single instance of pod nginx is running on each node of the Kubernetes cluster where nginx also represents the Image name which has to be used. Do not override any taints currently in place.

Use DaemonSet to complete this task and use ds-kusc00201 as DaemonSet name.

A . Explanation:

solution

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

root@node-1:~# vim ds.yaml

i

[Readme](#)[Web Terminal](#)

THE LINUX FOUNDATION

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      tolerations: []
        # this toleration is to have the daemonset runnable on master nodes
        # remove it if your masters can't run pods
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
        - name: nginx
          image: nginx
-- INSERT --
```

17,19

All

[Readme](#) >\_ [Web Terminal](#)

THE LINUX FOUNDATION

[Readme](#)

THE LINUX FOUNDATION

```
root@node-1:~# vim ds.yaml
iroot@node-1:~# k create -f ds.yaml
daemonset.apps/ds-kusc00201 created
root@node-1:~# k get ds
NAME      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
ds-kusc00201   2         2         2         2           2           <none>        4s
root@node-1:~# █
```

**Answer: A**

## Question: 66

List all persistent volumes sorted by capacity, saving the full kubectl output to /opt/KUCC00102/volume\_list. Use kubectl's own functionality for sorting the output, and do not manipulate it any further.

A . Explanation:

solution

The screenshot shows a terminal window with two tabs: "Readme" and "Web Terminal". The "Web Terminal" tab is active, displaying the output of a command. The command is "k get pv --sort-by=.spec.capacity.storage > /opt/KUCC00102/volume\_list". The output is a table of persistent volumes (PVs) with columns: PV name, Capacity, Access Mode, Recycle policy, Status, and I/O latency. The PVs are listed from smallest to largest capacity. All PVs have an access mode of "RWO" and a recycle policy of "Recycle". The status for most PVs is "Available", except for one which is "slow". The I/O latency for all PVs is also "slow".

PV Name	Capacity	Access Mode	Recycle	Status	I/O Latency
pv0007	7Gi	RWO	Recycle	Available	slow
pv0006	8Gi	RWO	Recycle	Available	slow
pv0003	10Gi	RWO	Recycle	Available	slow
pv0002	11Gi	RWO	Recycle	Available	slow
pv0010	13Gi	RWO	Recycle	Available	slow
pv0011	14Gi	RWO	Recycle	Available	slow
pv0001	16Gi	RWO	Recycle	Available	slow
pv0009	17Gi	RWO	Recycle	Available	slow
pv0005	18Gi	RWO	Recycle	Available	slow
pv0008	19Gi	RWO	Recycle	Available	slow
pv0000	21Gi	RWO	Recycle	Available	slow

```
root@node-1:~# k get pv --sort-by=.spec.capacity.storage > /opt/KUCC00102/volume_list
root@node-1:~#
```

---

**Answer: A**

---

## **Question: 67**

---

Monitor the logs of pod foo and:

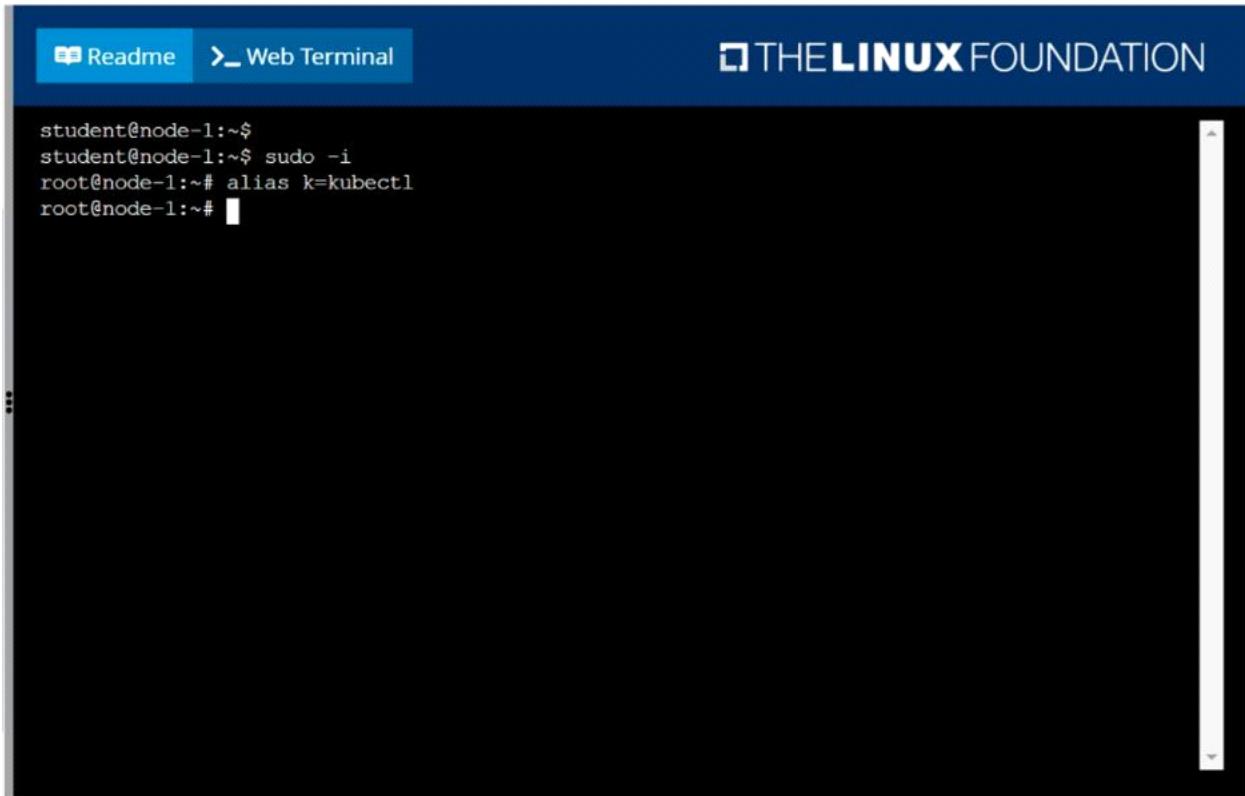
Extract log lines corresponding to error

unable-to-access-website

Write them to  
/opt/KULM00201/foo

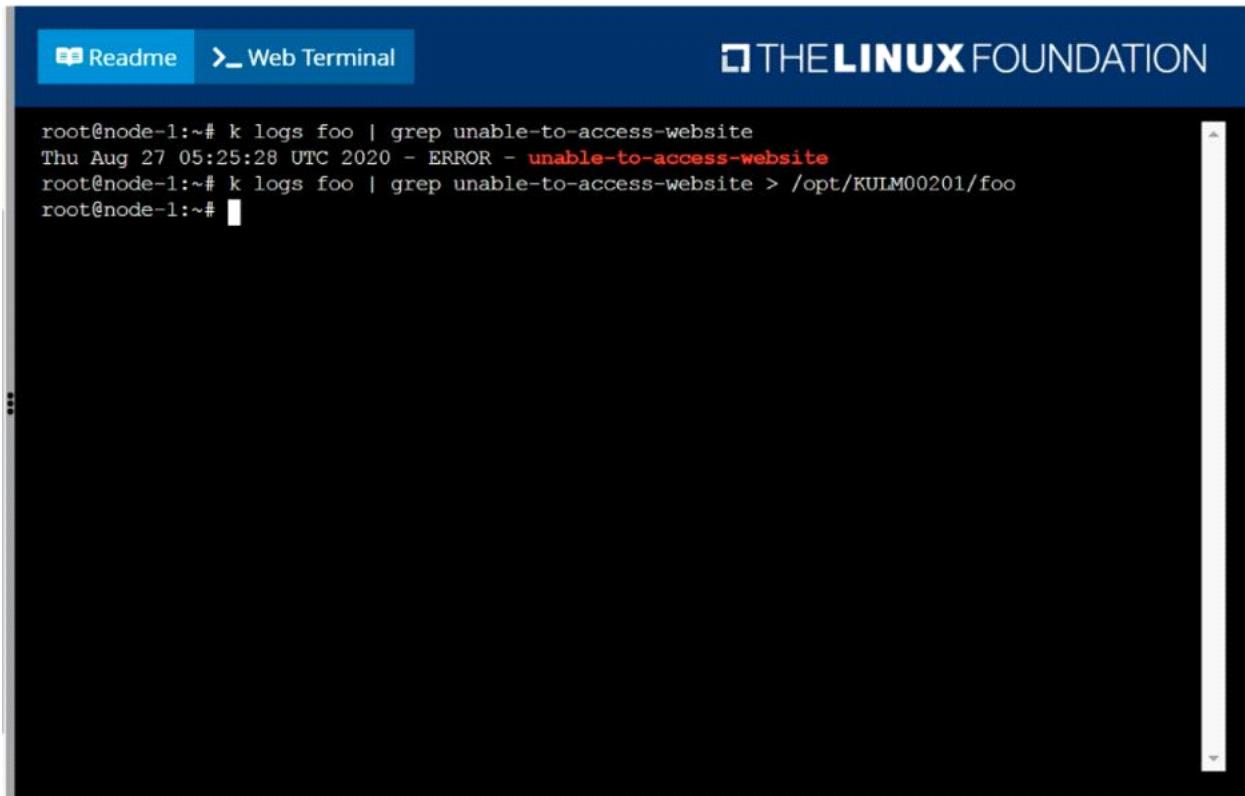


A . Explanation:  
solution



The screenshot shows a terminal window with a blue header bar. On the left of the header bar are two buttons: "Readme" and "Web Terminal". On the right of the header bar is the "THE LINUX FOUNDATION" logo. The main area of the terminal is black and contains the following text:

```
student@node-1:~$  
student@node-1:~$ sudo -i  
root@node-1:~# alias k=kubectl  
root@node-1:~#
```



The screenshot shows a terminal window with a blue header bar. On the left of the header bar are two buttons: "Readme" and "Web Terminal". On the right of the header bar is the "THE LINUX FOUNDATION" logo. The main area of the terminal is black and contains the following text:

```
root@node-1:~# k logs foo | grep unable-to-access-website  
Thu Aug 27 05:25:28 UTC 2020 - ERROR - unable-to-access-website  
root@node-1:~# k logs foo | grep unable-to-access-website > /opt/KULM00201/foo  
root@node-1:~#
```

---

Answer: A