



Free Questions for **CKAD** by **braindumpscollection**

Shared by **Roberts** on **06-06-2022**

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

Exhibit:



```
Set configuration context:   
[student@node-1] $ | kubectl  
config use-context k8s
```

Given a container that writes a log file in format A and a container that converts log files from format A to format B, create a deployment that runs both containers such that the log files from the first container are converted by the second container, emitting logs in format B.

Task:

- * Create a deployment named deployment-xyz in the default namespace, that:
- * Includes a primary lfcncf/busybox:1 container, named logger-dev
- * includes a sidecar lfcncf/fluentd:v0.12 container, named adapter-zen
- * Mounts a shared volume /tmp/log on both containers, which does not persist when the pod is deleted

* Instructs the logger-dev

container to run the command

```
while true; do
echo "i luv cncf" >> /
tmp/log/input.log;
sleep 10;
done
```

which should output logs to /tmp/log/input.log in plain text format, with example values:

```
i luv cncf
i luv cncf
i luv cncf
```

* The adapter-zen sidecar container should read /tmp/log/input.log and output the data to /tmp/log/output.* in Fluentd JSON format. Note that no knowledge of Fluentd is required to complete this task: all you will need to achieve this is to create the ConfigMap from the spec file provided at /opt/KDMC00102/fluentd-configmap.yaml , and mount that ConfigMap to /fluentd/etc in the adapter-zen sidecar container

Options:

A- Solution:

```
student@node-1:~$ kubectl create deployment deployment-xyz --image=lfcncf/busybox:1 --dry-run=c
lient -o yaml > deployment_xyz.yml
student@node-1:~$ vim deployment_xyz.yml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: deployment-xyz
  name: deployment-xyz
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deployment-xyz
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: deployment-xyz
    spec:
      containers:
      - image: lfcncf/busybox:1
        name: busybox
        resources: {}
status: {}
~
~
"deployment_xyz.yml" 24L, 434C
```

```
kind: Deployment
metadata:
  labels:
    app: deployment-xyz
    name: deployment-xyz
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deployment-xyz
  template:
    metadata:
      labels:
        app: deployment-xyz
    spec:
      volumes:
      - name: myvol1
        emptyDir: {}
      containers:
      - image: lfcncf/busybox:1
        name: logger-dev
        volumeMounts:
        - name: myvol1
          mountPath: /tmp/log
      - image: lfcncf/fluentd:v0.12
        name: adapter-zen
```

3 lines yanked

27,22

Bot

```
metadata:
  labels:
    app: deployment-xyz
spec:
  volumes:
  - name: myvol1
    emptyDir: {}
  - name: myvol2
    configMap:
      name: logconf
  containers:
  - image: lfcncf/busybox:1
    name: logger-dev
    command: ["/bin/sh", "-c", "while [ true ]; do echo 'i luv cncf' >> /tmp/log/input.log; sl
eep 10; done"]
    volumeMounts:
    - name: myvol1
      mountPath: /tmp/log
  - image: lfcncf/fluentd:v0.12
    name: adapter-zen
    command: ["/bin/sh", "-c", "tail -f /tmp/log/input.log >> /tmp/log/output.log"]
    volumeMounts:
    - name: myvol1
      mountPath: /tmp/log
    - name: myvol2
      mountPath: /fluentd/et
```

```
student@node-1:~$ kubectl create -f deployment_xyz.yml
deployment.apps/deployment-xyz created
student@node-1:~$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz 0/1     1            0           5s
student@node-1:~$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz 0/1     1            0           9s
student@node-1:~$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz 1/1     1            1           12s
student@node-1:~$
```

```
student@node-1:~$ kubectl create -f deployment_xyz.yml
deployment.apps/deployment-xyz created
student@node-1:~$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz 0/1     1            0           5s
student@node-1:~$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz 0/1     1            0           9s
student@node-1:~$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz 1/1     1            1           12s
student@node-1:~$
```

B- Solution:

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

THE **LINUX** FOUNDATION

```
student@node-1:~$ kubectl create deployment deployment-xyz --image=lfcncf/busybox:1 --dry-run=c
lient -o yaml > deployment_xyz.yml
student@node-1:~$ vim deployment_xyz.yml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: deployment-xyz
  name: deployment-xyz
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deployment-xyz
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: deployment-xyz
    spec:
      containers:
      - image: lfccncf/busybox:1
        name: busybox
        resources: {}
status: {}
~
~
"deployment_xyz.yml" 24L, 434C
```

3,1

All


```
kind: Deployment
metadata:
  labels:
    app: deployment-xyz
    name: deployment-xyz
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deployment-xyz
  template:
    metadata:
      labels:
        app: deployment-xyz
    spec:
      volumes:
      - name: myvol1
        emptyDir: {}
      containers:
      - image: lfcncf/busybox:1
        name: logger-dev
        volumeMounts:
        - name: myvol1
          mountPath: /tmp/log
      - image: lfcncf/fluentd:v0.12
        name: adapter-zen
```

3 lines yanked

27,22

Bot

```
metadata:
  labels:
    app: deployment-xyz
spec:
  volumes:
  - name: myvol1
    emptyDir: {}
  - name: myvol2
    configMap:
      name: logconf
  containers:
  - image: lfcncf/busybox:1
    name: logger-dev
    command: ["/bin/sh", "-c", "while [ true ]; do echo 'i luv cncf' >> /tmp/log/input.log; sl
eep 10; done"]
    volumeMounts:
    - name: myvol1
      mountPath: /tmp/log
  - image: lfcncf/fluentd:v0.12
    name: adapter-zen
    command: ["/bin/sh", "-c", "tail -f /tmp/log/input.log >> /tmp/log/output.log"]
    volumeMounts:
    - name: myvol1
      mountPath: /tmp/log
    - name: myvol2
      mountPath: /fluentd/et
```

```
student@node-1:~$ kubectl create -f deployment_xyz.yml
deployment.apps/deployment-xyz created
student@node-1:~$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz 0/1     1            0           5s
student@node-1:~$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz 0/1     1            0           9s
student@node-1:~$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz 1/1     1            1           12s
student@node-1:~$
```

Answer:

A

Question 2

Question Type: MultipleChoice

Exhibit:

Set configuration context:



```
[student@node-1] $ | kubectl config  
use-context sk8s
```

Context


A project that you are working on has a requirement for persistent data to be available.

Task


To facilitate this, perform the following tasks:

- * Create a file on node sk8s-node-0 at /opt/KDSP00101/data/index.html with the content Acct=Finance
- * Create a PersistentVolume named task-pv-volume using hostPath and allocate 1Gi to it, specifying that the volume is at /opt/KDSP00101/data on the cluster's node. The configuration should specify the access mode of ReadWriteOnce . It should define the StorageClass name exam for the PersistentVolume , which will be used to bind PersistentVolumeClaim requests to this PersistentVolume.

- * Create a PersistentVolumeClaim named task-pv-claim that requests a volume of at least 100Mi and specifies an access mode of ReadWriteOnce
- * Create a pod that uses the PersistentVolumeClaim as a volume with a label app: my-storage-app mounting the resulting volume to a mountPath /usr/share/nginx/html inside the pod

You can access `sk8s-node-0` by  issuing the following command:

```
[student@node-1] $ | ssh sk8s-node-0
```

Ensure that you return to the base node (with hostname `node-1`) once you have completed your work on `sk8s-node-0` 

Options:

A- Solution:

```
student@node-1:~$ kubectl config use-context sk8s
Switched to context "sk8s".
student@node-1:~$
```

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

System information as of Fri Oct 9 08:52:09 UTC 2020

System load: 2.02           Users logged in: 0
Usage of /: 10.3% of 242.29GB IP address for eth0: 10.250.3.115
Memory usage: 2%           IP address for docker0: 172.17.0.1
Swap usage: 0%             IP address for cni0: 10.244.1.1
Processes: 38

* Kubernetes 1.19 is out! Get it in one command with:

  sudo snap install microk8s --channel=1.19 --classic

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

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

New release '20.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

student@sk8s-node-0:~$
```

```
student@sk8s-node-0:~$ echo 'Acct=Finance' > /opt/KDSP00101/data/index.html
student@sk8s-node-0:~$ vim pv.yml
```

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

-- INSERT --

0,1

All

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: storage
  hostPath:
    path: /opt/KDSP00101/data
    type: Directory
```

```
student@sk8s-node-0:~$ kubectl create -f pv.yml
persistentvolume/task-pv-volume created
student@sk8s-node-0:~$ kubectl create -f pvc.yml
persistentvolumeclaim/task-pv-claim created
student@sk8s-node-0:~$ kubectl get pv
NAME                CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM                STORAGECLASS  AGE
task-pv-volume      1Gi       RWO           Retain          Bound   default/task-pv-claim  storage        11s
student@sk8s-node-0:~$ kubectl get pvc
NAME                STATUS  VOLUME             CAPACITY  ACCESS MODES  STORAGECLASS  AGE
task-pv-claim       Bound   task-pv-volume     1Gi       RWO           storage        9s
student@sk8s-node-0:~$ vim pod.yml
```



```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
  labels:
    app: my-storage-app
spec:
  containers:
  - name: myfrontend
    image: nginx
    volumeMounts:
    - mountPath: "/usr/share/nginx/html"
      name: mypod
  volumes:
  - name: mypod
    persistentVolumeClaim:
      claimName: task-pv-claim
~
~
~
~
~
~
~
~
```

17,32

All

```
student@sk8s-node-0:~$ kubectl create -f pod.yml
pod/mypod created
student@sk8s-node-0:~$ kubectl get
```

```
student@sk8s-node-0:~$ kubectl get pods
NAME      READY   STATUS             RESTARTS   AGE
mypod     0/1     ContainerCreating  0          4s
student@sk8s-node-0:~$ kubectl get pods
NAME      READY   STATUS             RESTARTS   AGE
mypod     0/1     ContainerCreating  0          8s
student@sk8s-node-0:~$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
mypod     1/1     Running   0          10s
student@sk8s-node-0:~$ logout
Connection to 10.250.3.115 closed.
student@node-1:~$
```

```
student@sk8s-node-0:~$ kubectl get pods
NAME      READY   STATUS             RESTARTS   AGE
mypod     0/1     ContainerCreating  0          4s
student@sk8s-node-0:~$ kubectl get pods
NAME      READY   STATUS             RESTARTS   AGE
mypod     0/1     ContainerCreating  0          8s
student@sk8s-node-0:~$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
mypod     1/1     Running   0          10s
student@sk8s-node-0:~$ logout
Connection to 10.250.3.115 closed.
student@node-1:~$
```

B- Solution:

```
student@node-1:~$ kubectl config use-context sk8s
Switched to context "sk8s".
student@node-1:~$
```

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

System information as of Fri Oct 9 08:52:09 UTC 2020

System load: 2.02           Users logged in: 0
Usage of /: 10.3% of 242.29GB IP address for eth0: 10.250.3.115
Memory usage: 2%           IP address for docker0: 172.17.0.1
Swap usage: 0%             IP address for cni0: 10.244.1.1
Processes: 38

* Kubernetes 1.19 is out! Get it in one command with:

  sudo snap install microk8s --channel=1.19 --classic

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

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

New release '20.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

student@sk8s-node-0:~$
```

```
student@sk8s-node-0:~$ echo 'Acct=Finance' > /opt/KDSP00101/data/index.html
student@sk8s-node-0:~$ vim pv.yml
```

-- INSERT --

0,1

All

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: storage
  hostPath:
    path: /opt/KDSP00101/data
    type: Directory
```

```
student@sk8s-node-0:~$ kubectl create -f pv.yml
persistentvolume/task-pv-volume created
student@sk8s-node-0:~$ kubectl create -f pvc.yml
persistentvolumeclaim/task-pv-claim created
student@sk8s-node-0:~$ kubectl get pv
NAME                CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM                STORAGECLASS  AGE
task-pv-volume      1Gi       RWO           Retain          Bound   default/task-pv-claim  storage        11s
student@sk8s-node-0:~$ kubectl get pvc
NAME                STATUS  VOLUME             CAPACITY  ACCESS MODES  STORAGECLASS  AGE
task-pv-claim       Bound   task-pv-volume     1Gi       RWO           storage        9s
student@sk8s-node-0:~$ vim pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
  labels:
    app: my-storage-app
spec:
  containers:
  - name: myfrontend
    image: nginx
    volumeMounts:
    - mountPath: "/usr/share/nginx/html"
      name: mypod
  volumes:
  - name: mypod
    persistentVolumeClaim:
      claimName: task-pv-claim
~
~
~
~
~
~
~
~
```

17,32

All

```
student@sk8s-node-0:~$ kubectl create -f pod.yml
pod/mypod created
student@sk8s-node-0:~$ kubectl get
```

```
student@sk8s-node-0:~$ kubectl get pods
NAME     READY   STATUS             RESTARTS   AGE
mypod    0/1     ContainerCreating   0           4s
student@sk8s-node-0:~$ kubectl get pods
NAME     READY   STATUS             RESTARTS   AGE
mypod    0/1     ContainerCreating   0           8s
student@sk8s-node-0:~$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
mypod    1/1     Running   0           10s
student@sk8s-node-0:~$ logout
Connection to 10.250.3.115 closed.
student@node-1:~$
```

Answer:

A

Question 3

Question Type: MultipleChoice

Exhibit:

Set configuration context:



```
[student@node-1] $ | kubectl config  
use-context dk8s
```

Context

A user has reported an application is unteachable due to a failing livenessProbe .

Task

Perform the following tasks:

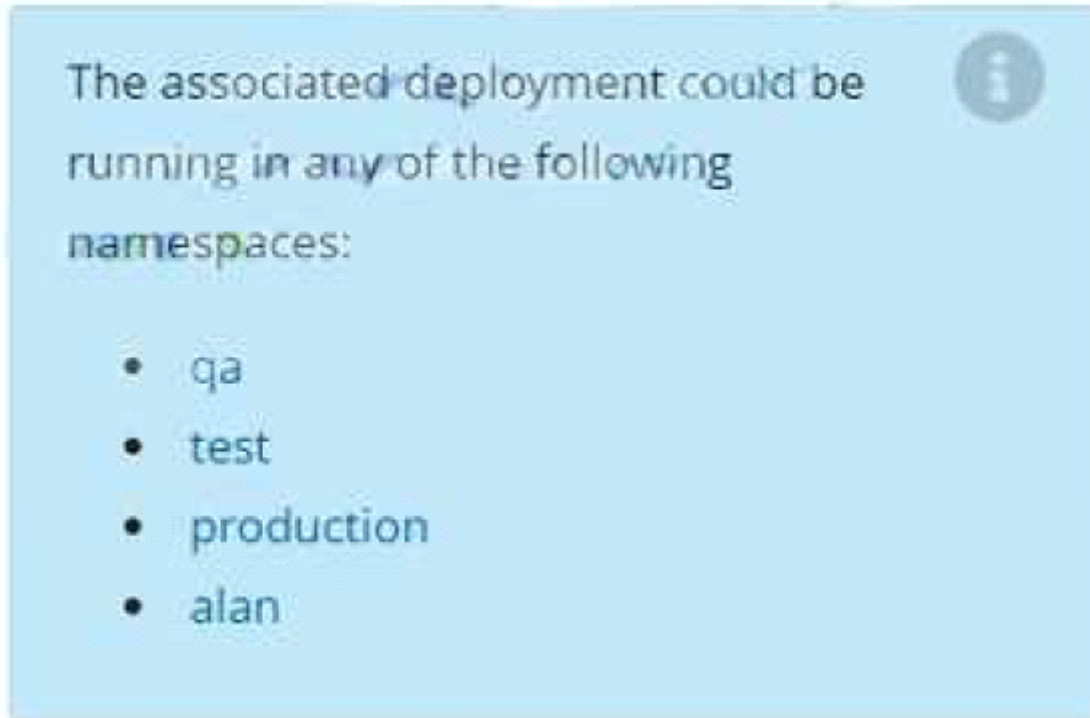
* Find the broken pod and store its name and namespace to /opt/KDOB00401/broken.txt in the format:

```
<namespace>/<pod>
```

The output file has already been created

* Store the associated error events to a file /opt/KDOB00401/error.txt, The output file has already been created. You will need to use the -o wide output specifier with your command

* Fix the issue.



Options:

A- Solution:

Create the Pod:

```
kubectl create -f http://k8s.io/docs/tasks/configure-pod-container/exec-liveness.yaml
```

Within 30 seconds, view the Pod events:

```
kubectl describe pod liveness-exec
```

The output indicates that no liveness probes have failed yet:

```
FirstSeen LastSeen Count From SubobjectPath Type Reason Message
```

```
-----
```

```
24s 24s 1 {default-scheduler } Normal Scheduled Successfully assigned liveness-exec to worker0
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image 'gcr.io/google_containers/busybox'
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image 'gcr.io/google_containers/busybox'
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Created Created container with docker id 86849c15382e;
Security:[seccomp=unconfined]
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Started Started container with docker id 86849c15382e
```

After 35 seconds, view the Pod events again:

```
kubectl describe pod liveness-exec
```

At the bottom of the output, there are messages indicating that the liveness probes have failed, and the containers have been killed and recreated.

```
FirstSeen LastSeen Count From SubobjectPath Type Reason Message
```

```
-----
```

```
37s 37s 1 {default-scheduler } Normal Scheduled Successfully assigned liveness-exec to worker0
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image 'gcr.io/google_containers/busybox'
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image 'gcr.io/google_containers/busybox'
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Created Created container with docker id 86849c15382e;
Security:[seccomp=unconfined]
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Started Started container with docker id 86849c15382e
2s 2s 1 {kubelet worker0} spec.containers{liveness} Warning Unhealthy Liveness probe failed: cat: can't open '/tmp/healthy': No such file
or directory
```

Wait another 30 seconds, and verify that the Container has been restarted:

```
kubectl get pod liveness-exec
```

The output shows that `RESTARTS` has been incremented:

```
NAME READY STATUS RESTARTS AGE
```

```
liveness-exec 1/1 Running 1 m
```

B- Solution:

Create the Pod:

```
kubectl create -f http://k8s.io/docs/tasks/configure-pod-container/exec-liveness.yaml
```

Within 30 seconds, view the Pod events:

```
kubectl describe pod liveness-exec
```

The output indicates that no liveness probes have failed yet:

```
FirstSeen LastSeen Count From SubobjectPath Type Reason Message
```

```
-----
```

```
24s 24s 1 {default-scheduler } Normal Scheduled Successfully assigned liveness-exec to worker0
```

```
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image 'gcr.io/google_containers/busybox'
```

```
kubectl describe pod liveness-exec
```

At the bottom of the output, there are messages indicating that the liveness probes have failed, and the containers have been killed and recreated.

```
FirstSeen LastSeen Count From SubobjectPath Type Reason Message
```

```
-----
```

```
37s 37s 1 {default-scheduler } Normal Scheduled Successfully assigned liveness-exec to worker0
```

```
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image 'gcr.io/google_containers/busybox'
```

```
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image 'gcr.io/google_containers/busybox'
```

```
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Created Created container with docker id 86849c15382e;
```

```
Security:[seccomp=unconfined]
```

```
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Started Started container with docker id 86849c15382e
2s 2s 1 {kubelet worker0} spec.containers{liveness} Warning Unhealthy Liveness probe failed: cat: can't open '/tmp/healthy': No such file
or directory
```

Wait another 30 seconds, and verify that the Container has been restarted:

```
kubectl get pod liveness-exec
```

The output shows that RESTARTS has been incremented:

```
NAME READY STATUS RESTARTS AGE
```

```
liveness-exec 1/1 Running 1 m
```

Answer:

A

Question 4

Question Type: MultipleChoice

Exhibit:

Set configuration context:



```
[student@node-1] $ | kubectl config  
use-context nk8s
```

Task

You have rolled out a new pod to your infrastructure and now you need to allow it to communicate with the web and storage pods but nothing else. Given the running pod `kdsn00201 -newpod` edit it to use a network policy that will allow it to send and receive traffic only to and from the web and storage pods.

All work on this item should be
conducted in the `kdsn00201`
namespace.



All required NetworkPolicy resources are already created and ready for use as appropriate. You should not create, modify or delete any network policies whilst completing this item.



Options:

A- Pending

Answer:

A

Question 5

Question Type: MultipleChoice

Exhibit:

A terminal window with an orange header bar. The header bar contains the text "Set configuration context:" on the left and a warning triangle icon on the right. Below the header bar is a light blue rectangular area containing a terminal prompt and a command. The prompt is "[student@node-1] \$" and the command is "kubectl config use-context nk8s".

```
Set configuration context: [!]  
  
[student@node-1] $ | kubectl config use-context nk8s
```

Task

A deployment is falling on the cluster due to an incorrect image being specified. Locate the deployment, and fix the problem.

Options:

A- Pending

Answer:

A

Question 6

Question Type: MultipleChoice

Exhibit:



```
Set configuration context: [!]  
  
[student@node-1] $ | kubectl config  
use-context k8s
```

Context

Developers occasionally need to submit pods that run periodically.

Task

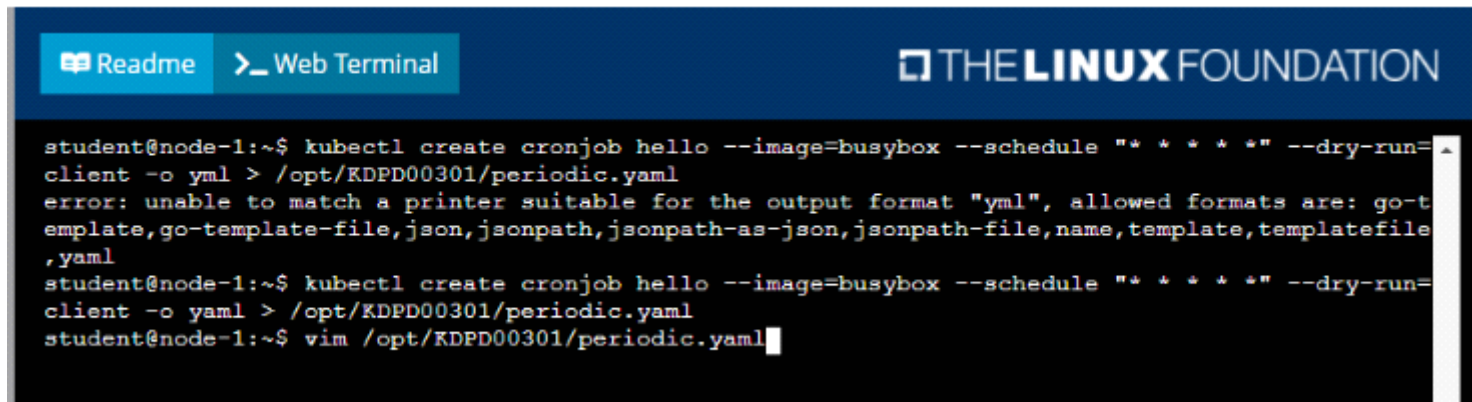
Follow the steps below to create a pod that will start at a predetermined time and]which runs to completion only once each time it is started:

* Create a YAML formatted Kubernetes manifest `/opt/KDPD00301/periodic.yaml` that runs the following shell command: `date` in a single busybox container. The command should run every minute and must complete within 22 seconds or be terminated by Kubernetes. The Cronjob name and container name should both be `hello`

* Create the resource in the above manifest and verify that the job executes successfully at least once

Options:

A- Solution:



The screenshot shows a web terminal interface with a dark blue header. On the left, there are two buttons: "Readme" and "Web Terminal". On the right, the logo for "THE LINUX FOUNDATION" is visible. The terminal content shows a user at a prompt `student@node-1:~$` running the command `kubect1 create cronjob hello --image=busybox --schedule "* * * * *" --dry-run=client -o yaml > /opt/KDPD00301/periodic.yaml`. The output is an error message: `error: unable to match a printer suitable for the output format "yaml", allowed formats are: go-template, go-template-file, json, jsonpath, jsonpath-as-json, jsonpath-file, name, template, templatefile, yaml`. The user then runs the same command again, and the output is the same error. Finally, the user runs `vim /opt/KDPD00301/periodic.yaml` at the prompt.

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  jobTemplate:
    metadata:
      name: hello
    spec:
      template:
        spec:
          containers:
            - image: busybox
              name: hello
              args: ["/bin/sh", "-c", "date"]
              restartPolicy: Never
  schedule: '* * * * *'
  startingDeadlineSeconds: 22
  concurrencyPolicy: Allow
```

```
~
~
~
~
~
~
```

```
student@node-1:~$ kubectl create cronjob hello --image=busybox --schedule "* * * * *" --dry-run=
client -o yaml > /opt/KDPD00301/periodic.yaml
error: unable to match a printer suitable for the output format "yaml", allowed formats are: go-t
emplate,go-template-file,json,jsonpath,jsonpath-as-json,jsonpath-file,name,template,templatefile
,yaml
student@node-1:~$ kubectl create cronjob hello --image=busybox --schedule "* * * * *" --dry-run=
client -o yaml > /opt/KDPD00301/periodic.yaml
student@node-1:~$ vim /opt/KDPD00301/periodic.yaml
student@node-1:~$ kubectl create -f /opt/KDPD00301/periodic.yaml
cronjob.batch/hello created
student@node-1:~$ kubectl get cronjob
NAME          SCHEDULE          SUSPEND   ACTIVE   LAST SCHEDULE   AGE
hello        */1 * * * *      False    0        <none>          6s
student@node-1:~$
```

B- Solution:

```
student@node-1:~$ kubectl create cronjob hello --image=busybox --schedule "* * * * *" --dry-run=
client -o yaml > /opt/KDPD00301/periodic.yaml
error: unable to match a printer suitable for the output format "yaml", allowed formats are: go-t
emplate,go-template-file,json,jsonpath,jsonpath-as-json,jsonpath-file,name,template,templatefile
,yaml
student@node-1:~$ kubectl create cronjob hello --image=busybox --schedule "* * * * *" --dry-run=
client -o yaml > /opt/KDPD00301/periodic.yaml
student@node-1:~$ vim /opt/KDPD00301/periodic.yaml
```

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  jobTemplate:
    metadata:
      name: hello
    spec:
      template:
        spec:
          containers:
            - image: busybox
              name: hello
              args: ["/bin/sh", "-c", "date"]
              restartPolicy: Never
  schedule: '* * * * *'
  startingDeadlineSeconds: 22
  concurrencyPolicy: Allow
```

Answer:

A

To Get Premium Files for CKAD Visit

<https://www.p2pexams.com/products/ckad>

For More Free Questions Visit

<https://www.p2pexams.com/linux-foundation/pdf/ckad>

