CKAD$^{Q\&As}$

Certified Kubernetes Application Developer (CKAD) Program

Pass Linux Foundation CKAD Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

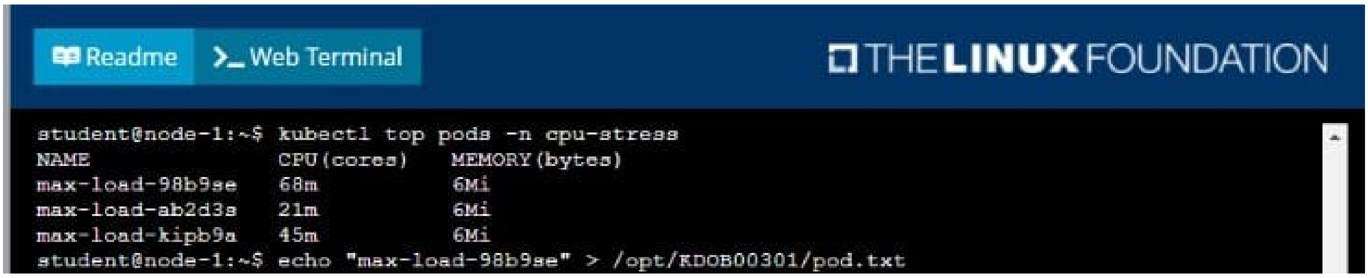
https://www.geekcert.com/ckad.html

100% Passing Guarantee
100% Money Back Assurance

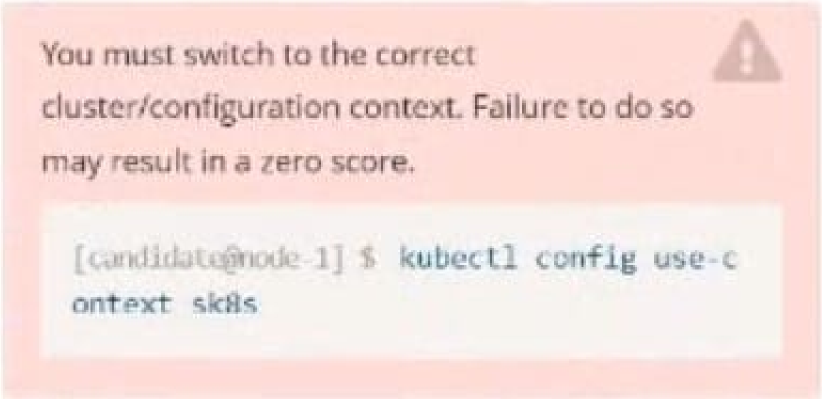
Following Questions and Answers are all new published by Linux Foundation Official Exam Center

QUESTION 1

CORRECT TEXT



```
student@node-1:~$ kubectl top pods -n cpu-stress
NAME                CPU(cores)    MEMORY(bytes)
max-load-98b9se     68m           6Mi
max-load-ab2d3s     21m           6Mi
max-load-kipb9a     45m           6Mi
student@node-1:~$ echo "max-load-98b9se" > /opt/KDOB00301/pod.txt
```



You must switch to the correct cluster/configuration context. Failure to do so may result in a zero score.

```
[candidate@node-1] $ kubectl config use-c
ontext sk8s
```

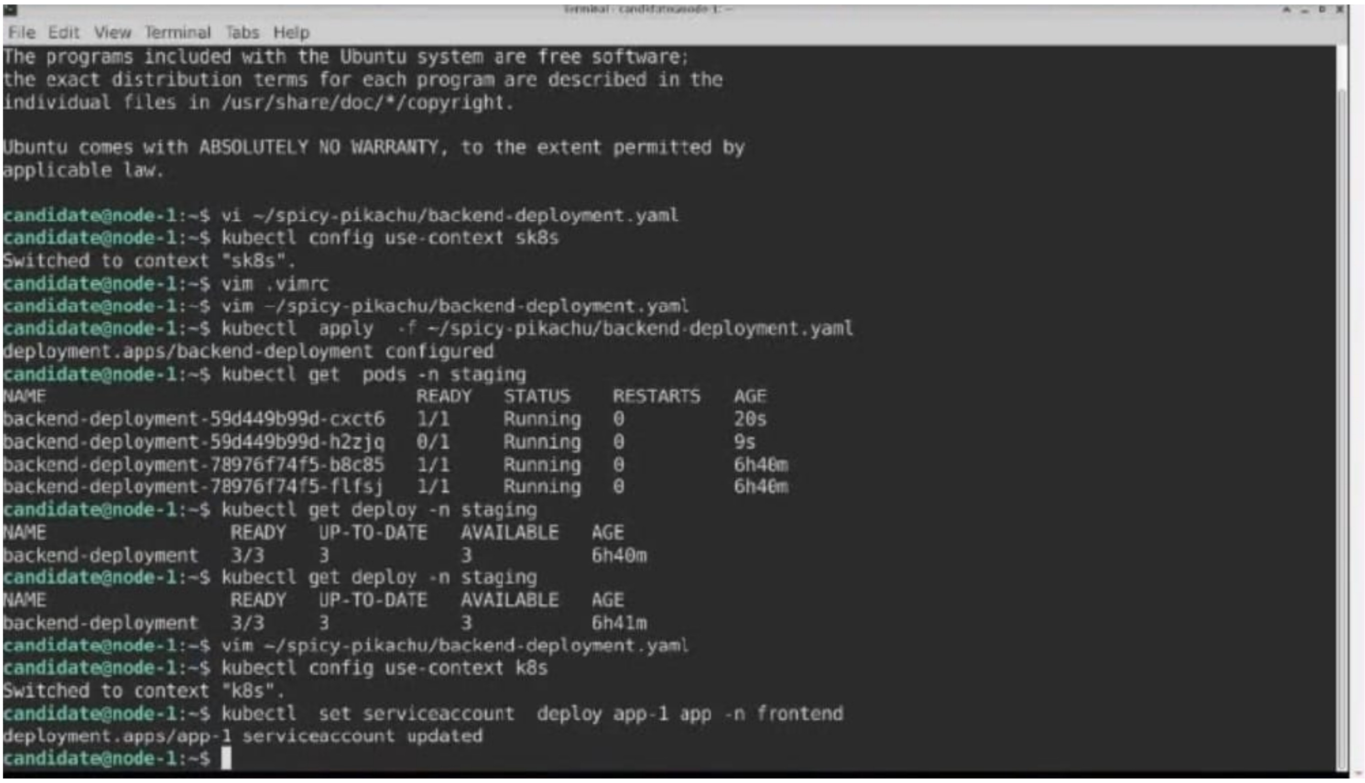
Task:

Update the Deployment app-1 in the frontend namespace to use the existing ServiceAccount app.

A. Please check explanations

B. Place Holder

Correct Answer: A



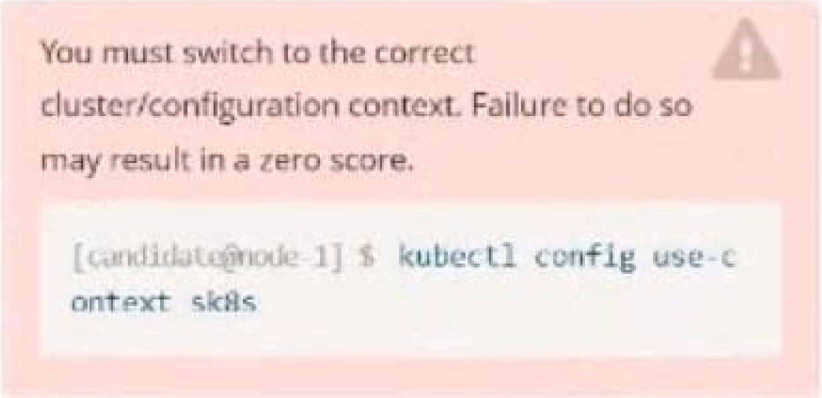
```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

candidate@node-1:~$ vi ~/spicy-pikachu/backend-deployment.yaml
candidate@node-1:~$ kubectl config use-context sk8s
Switched to context "sk8s".
candidate@node-1:~$ vim .vimrc
candidate@node-1:~$ vim ~/spicy-pikachu/backend-deployment.yaml
candidate@node-1:~$ kubectl  apply  -f ~/spicy-pikachu/backend-deployment.yaml
deployment.apps/backend-deployment configured
candidate@node-1:~$ kubectl get  pods -n staging
NAME                                 READY   STATUS    RESTARTS   AGE
backend-deployment-59d449b99d-cxct6   1/1     Running   0          20s
backend-deployment-59d449b99d-h2zjq   0/1     Running   0          9s
backend-deployment-78976f74f5-b8c85   1/1     Running   0          6h40m
backend-deployment-78976f74f5-flfsj   1/1     Running   0          6h40m
candidate@node-1:~$ kubectl get deploy -n staging
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
backend-deployment   3/3     3            3           6h40m
candidate@node-1:~$ kubectl get deploy -n staging
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
backend-deployment   3/3     3            3           6h41m
candidate@node-1:~$ vim ~/spicy-pikachu/backend-deployment.yaml
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl  set serviceaccount  deploy app-1 app -n frontend
deployment.apps/app-1 serviceaccount updated
candidate@node-1:~$ 
```

QUESTION 2

CORRECT TEXT



You must switch to the correct cluster/configuration context. Failure to do so may result in a zero score.

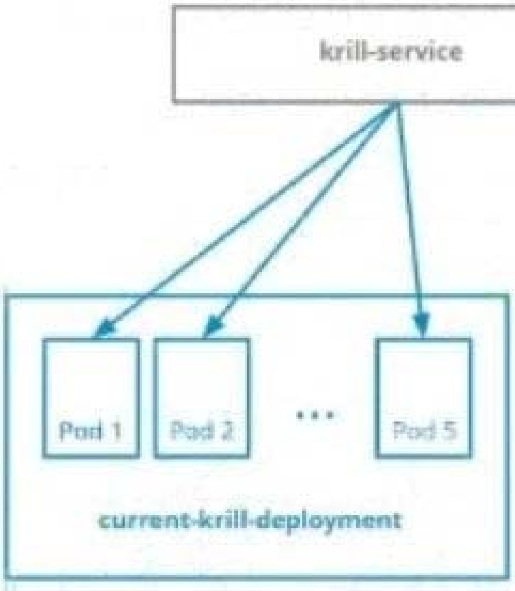
```
[candidate@node-1] $ kubectl config use-context sk8s
```

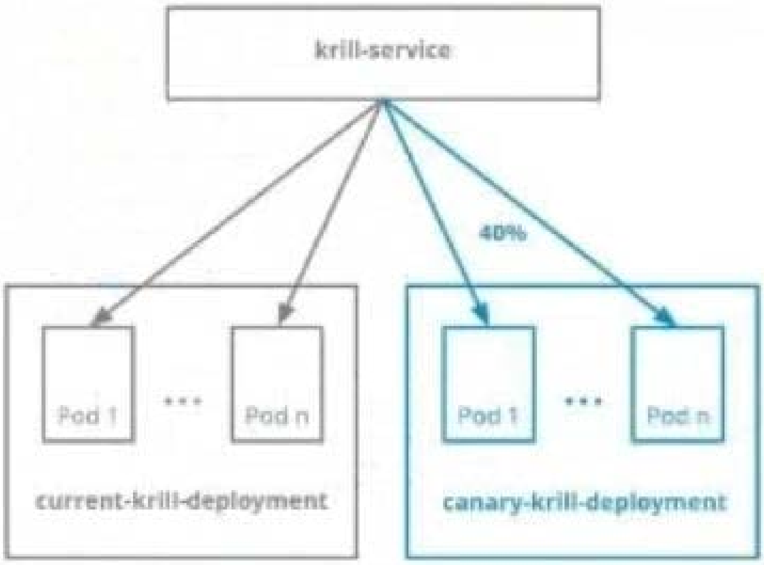
Context

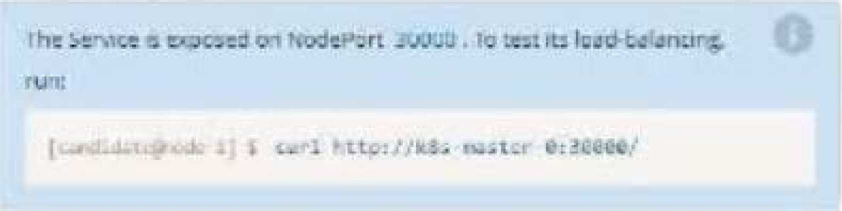
You are asked to prepare a Canary deployment for testing a new application release.

Task:

A Service named krill-Service in the goshark namespace points to 5 pod created by the Deployment named current-krill-deployment







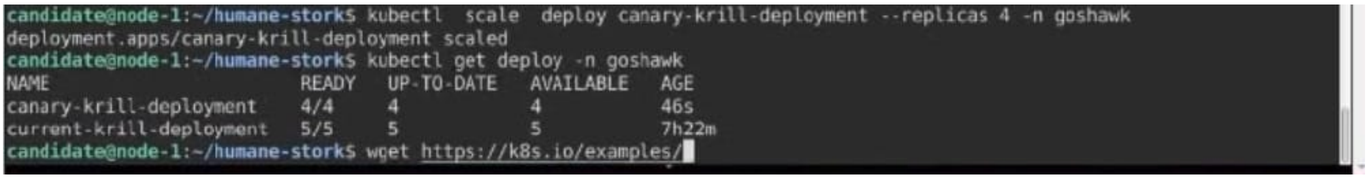
The Service is exposed on NodePort 30000 . To test its load-balancing, run:

```
[candidate@node-1] $ curl http://k8s-master-0:30000/
```

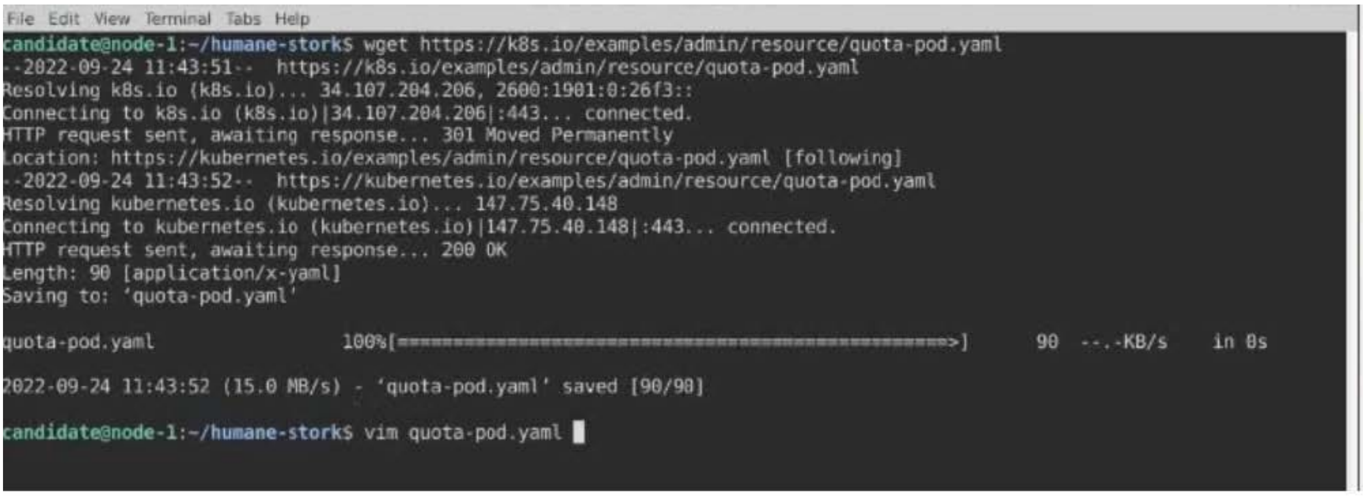
A. Please check explanations

B. Place Holder

Correct Answer: A



```
candidate@node-1:~/humane-stork$ kubectl  scale  deploy canary-krill-deployment --replicas 4 -n goshawk
deployment.apps/canary-krill-deployment scaled
candidate@node-1:~/humane-stork$ kubectl get deploy -n goshawk
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
canary-krill-deployment   4/4     4            4           46s
current-krill-deployment  5/5     5            5           7h22m
candidate@node-1:~/humane-stork$ wget https://k8s.io/examples/
```

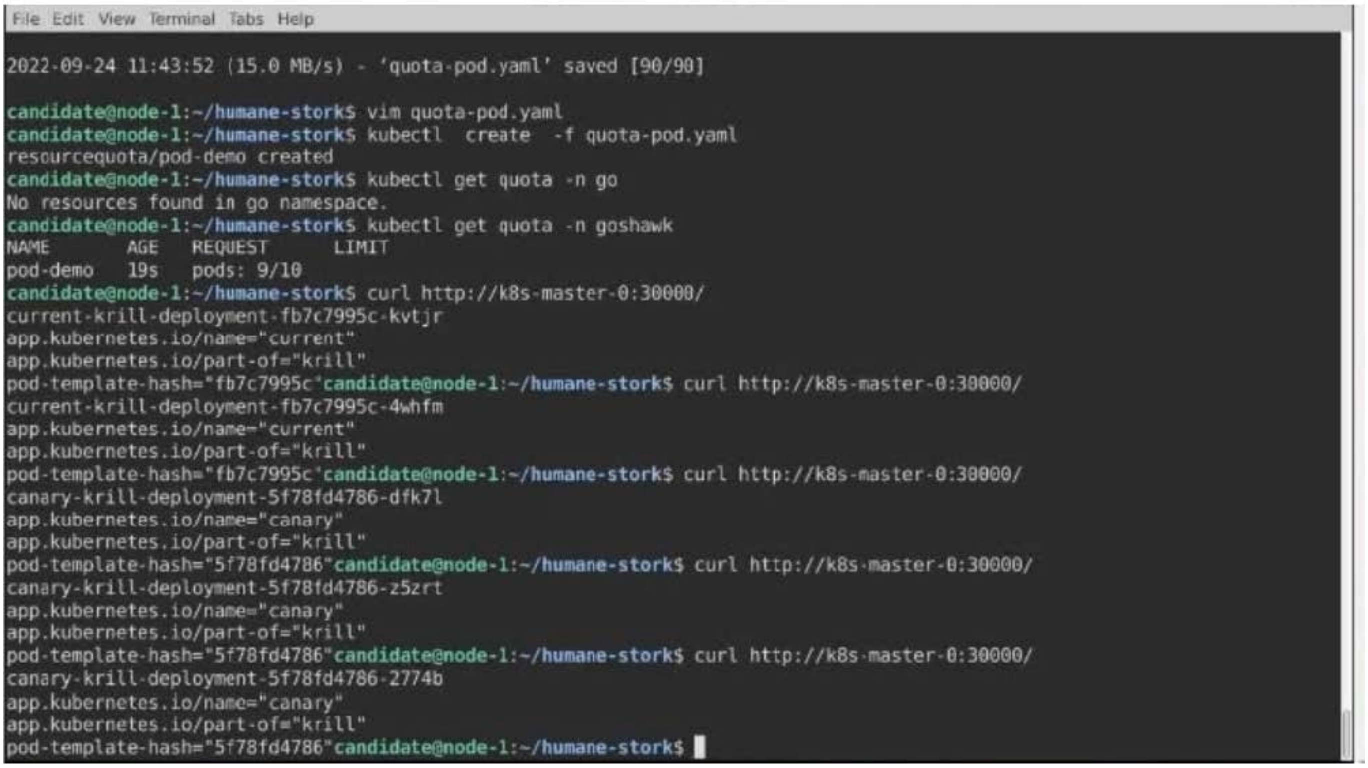


```
candidate@node-1:~/humane-stork$ wget https://k8s.io/examples/admin/resource/quota-pod.yaml
--2022-09-24 11:43:51--  https://k8s.io/examples/admin/resource/quota-pod.yaml
Resolving k8s.io (k8s.io)... 34.107.204.206, 2600:1901:0:26f3::
Connecting to k8s.io (k8s.io)|34.107.204.206|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://kubernetes.io/examples/admin/resource/quota-pod.yaml [following]
--2022-09-24 11:43:52--  https://kubernetes.io/examples/admin/resource/quota-pod.yaml
Resolving kubernetes.io (kubernetes.io)... 147.75.40.148
Connecting to kubernetes.io (kubernetes.io)|147.75.40.148|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 90 [application/x-yaml]
Saving to: 'quota-pod.yaml'

quota-pod.yaml              100%[=====================================================>]      90  --.-KB/s    in 0s

2022-09-24 11:43:52 (15.0 MB/s) - 'quota-pod.yaml' saved [90/90]

candidate@node-1:~/humane-stork$ vim quota-pod.yaml
```

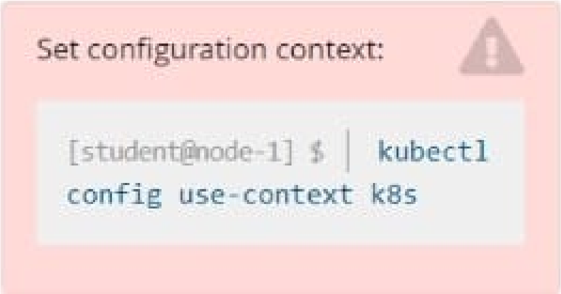


```
2022-09-24 11:43:52 (15.0 MB/s) - 'quota-pod.yaml' saved [90/90]

candidate@node-1:~/humane-stork$ vim quota-pod.yaml
candidate@node-1:~/humane-stork$ kubectl  create  -f quota-pod.yaml
resourcequota/pod-demo created
candidate@node-1:~/humane-stork$ kubectl get quota -n go
No resources found in go namespace.
candidate@node-1:~/humane-stork$ kubectl get quota -n goshawk
NAME       AGE   REQUEST      LIMIT
pod-demo   19s   pods: 9/10
candidate@node-1:~/humane-stork$ curl http://k8s-master-0:30000/
current-krill-deployment-fb7c7995c-kvtjr
app.kubernetes.io/name="current"
app.kubernetes.io/part-of="krill"
pod-template-hash="fb7c7995c"candidate@node-1:~/humane-stork$ curl http://k8s-master-0:30000/
current-krill-deployment-fb7c7995c-4whfm
app.kubernetes.io/name="current"
app.kubernetes.io/part-of="krill"
pod-template-hash="fb7c7995c"candidate@node-1:~/humane-stork$ curl http://k8s-master-0:30000/
canary-krill-deployment-5f78fd4786-dfk7l
app.kubernetes.io/name="canary"
app.kubernetes.io/part-of="krill"
pod-template-hash="5f78fd4786"candidate@node-1:~/humane-stork$ curl http://k8s-master-0:30000/
canary-krill-deployment-5f78fd4786-z5zrt
app.kubernetes.io/name="canary"
app.kubernetes.io/part-of="krill"
pod-template-hash="5f78fd4786"candidate@node-1:~/humane-stork$ curl http://k8s-master-0:30000/
canary-krill-deployment-5f78fd4786-2774b
app.kubernetes.io/name="canary"
app.kubernetes.io/part-of="krill"
pod-template-hash="5f78fd4786"candidate@node-1:~/humane-stork$
```

QUESTION 3

CORRECT TEXT



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

Task:

1.

Create a deployment named deployment-xyz in the default namespace, that:

2.

Includes a primary

3.

lfccncf/busybox:1 container, named logger-dev

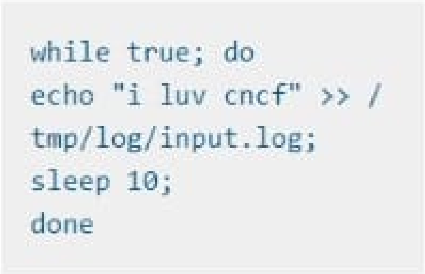
4.

Includes a sidecar Ifccncf/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

5.

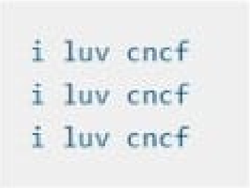
Instructs the logger-dev

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:



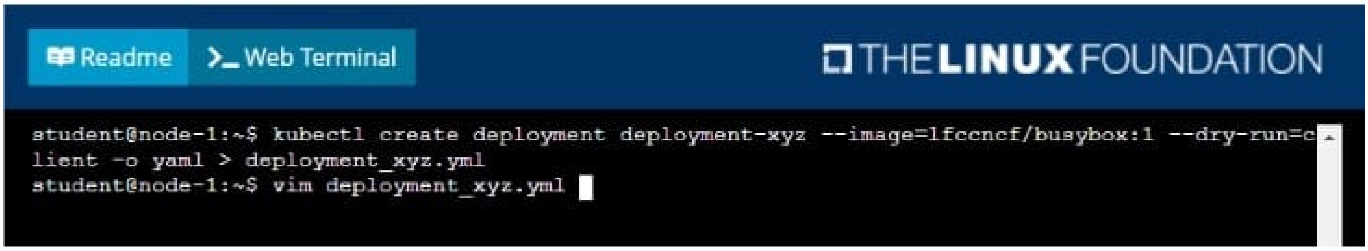
```
i luv cncf
i luv cncf
i luv cncf
```

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

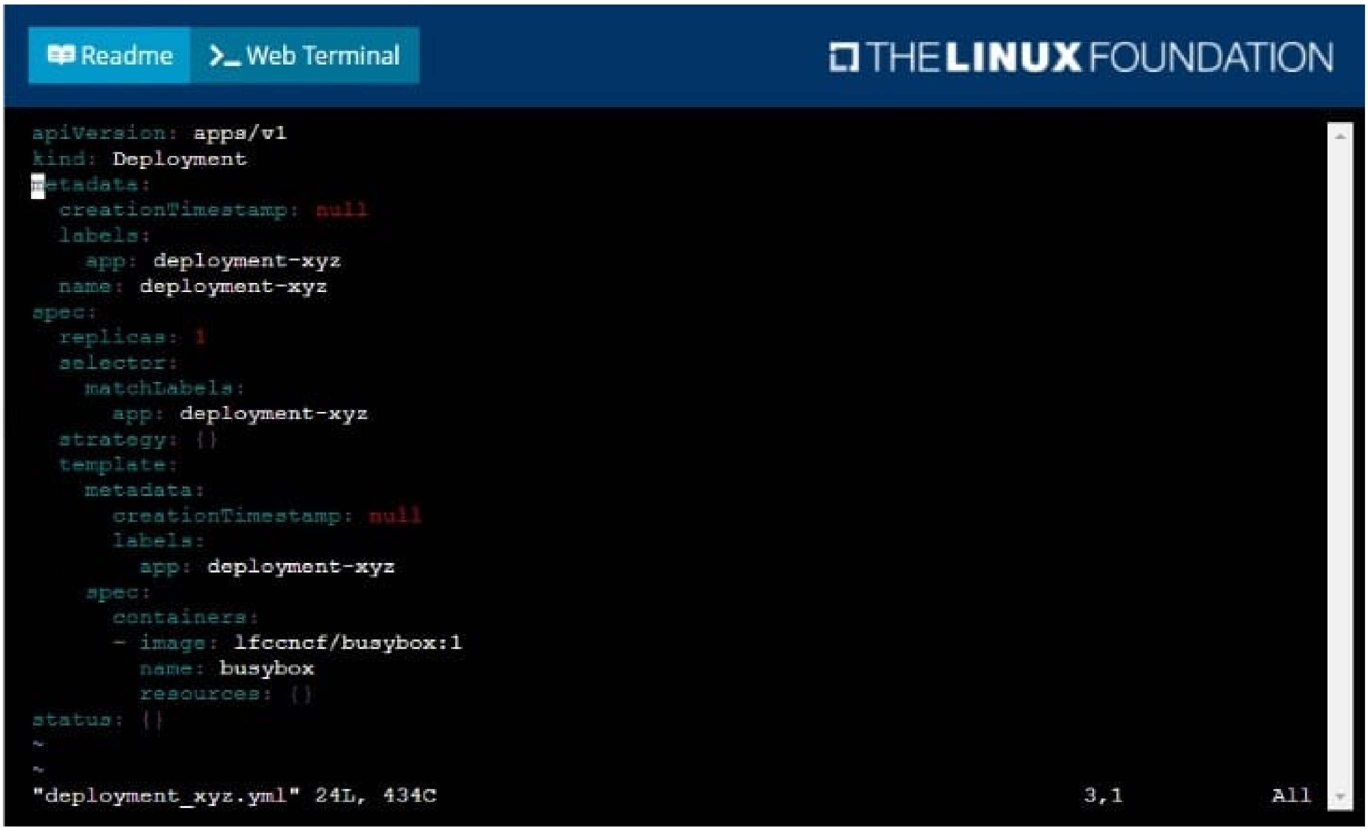
A. Please check explanations

B. Place Holder

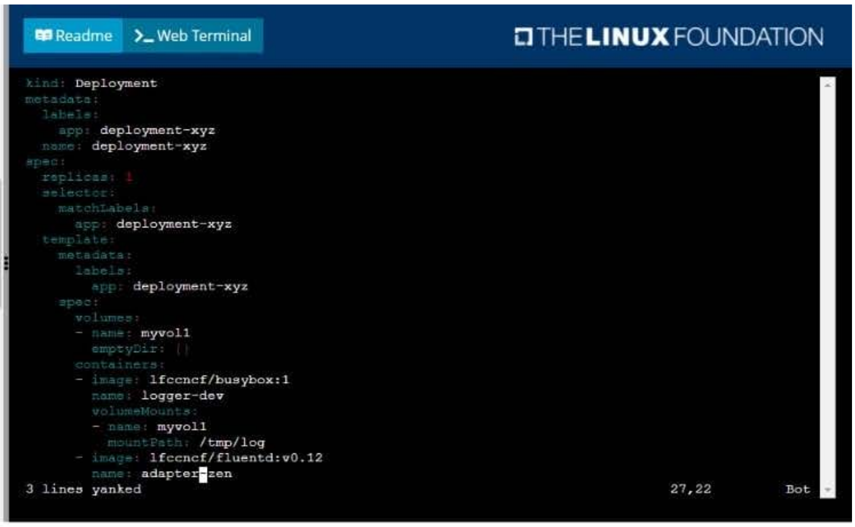
Correct Answer: A



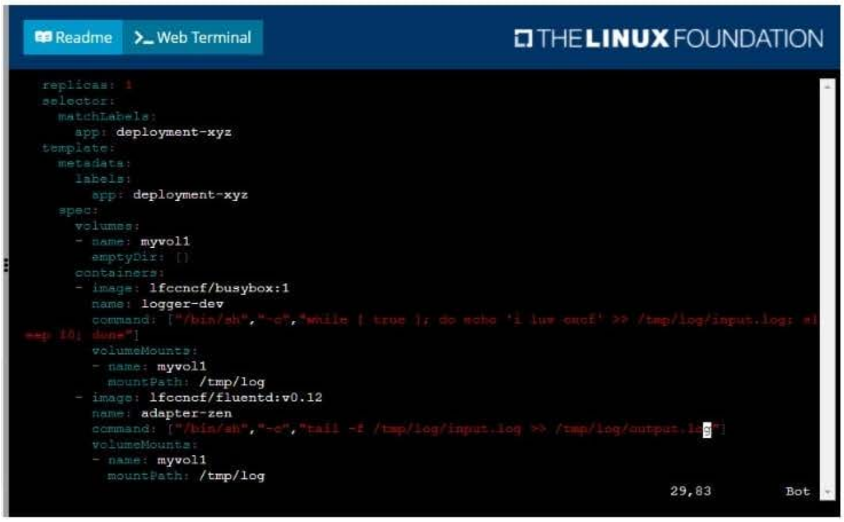
```
student@node-1:~$ kubectl create deployment deployment-xyz --image=lfccncf/busybox:1 --dry-run=c
lient -o yaml > deployment_xyz.yml
student@node-1:~$ vim deployment_xyz.yml
```



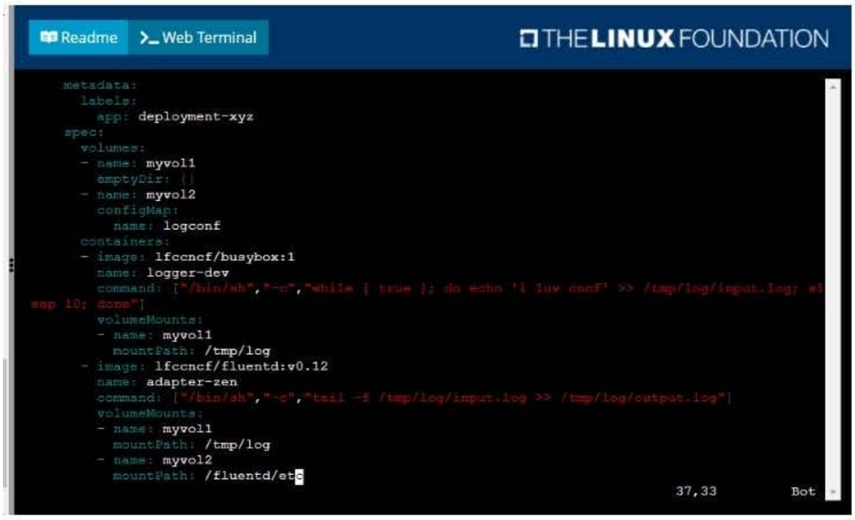
```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: deployment-xyz
  name: deployment-xyz
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deployment-xyz
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: deployment-xyz
    spec:
      containers:
      - image: lfccncf/busybox:1
        name: busybox
        resources: {}
status: {}
~
~
"deployment_xyz.yml" 24L, 434C                                          3,1           All
```



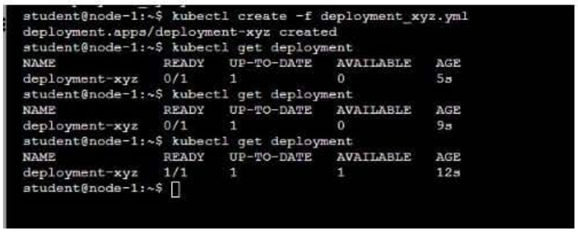
```
kind: Deployment
metadata:
  labels:
    app: deployment-xyz
  name: deployment-xyz
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deployment-xyz
  template:
    metadata:
      labels:
        app: deployment-xyz
    spec:
      volumes:
      - name: myvol1
        emptyDir: {}
      containers:
      - image: lfccncf/busybox:1
        name: logger-dev
        volumeMounts:
        - name: myvol1
          mountPath: /tmp/log
      - image: lfccncf/fluentd:v0.12
        name: adapter-zen
3 lines yanked                                                    27,22          Bot
```



```
  replicas: 1
  selector:
    matchLabels:
      app: deployment-xyz
  template:
    metadata:
      labels:
        app: deployment-xyz
    spec:
      volumes:
      - name: myvol1
        emptyDir: {}
      containers:
      - image: lfccncf/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: lfccncf/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
                                                                  29,83          Bot
```



```
    metadata:
      labels:
        app: deployment-xyz
    spec:
      volumes:
      - name: myvol1
        emptyDir: {}
      - name: myvol2
        configMap:
          name: logconf
      containers:
      - image: lfccncf/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: lfccncf/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/etc
                                                                  37,33          Bot
```



```
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:~$
```

QUESTION 4

CORRECT TEXT Context Anytime a team needs to run a container on Kubernetes they will need to define a pod within which to run the container. Task Please complete the following:

1.

Create a YAML formatted pod manifest

2.

/opt/KDPD00101/podl.yml to create a pod named app1 that runs a container named app1cont using image lfccncf/arg-output with these command line arguments: -lines 56 -F

3.

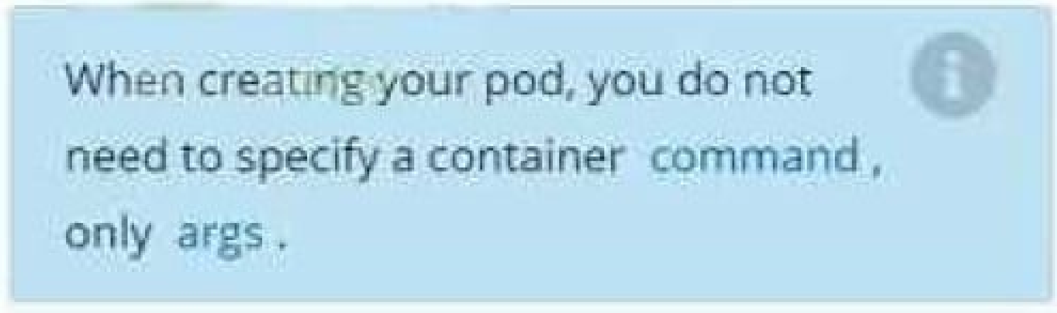
Create the pod with the kubect1 command using the YAML file created in the previous step

4.

When the pod is running display summary data about the pod in JSON format using the kubect1 command and redirect the output to a file named /opt/KDPD00101/out1.json

5.

All of the files you need to work with have been created, empty, for your convenience

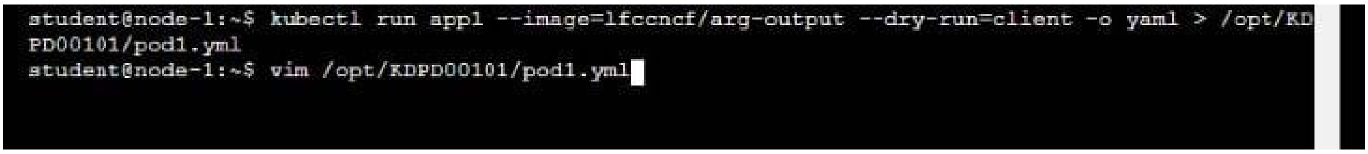


When creating your pod, you do not need to specify a container command, only args.

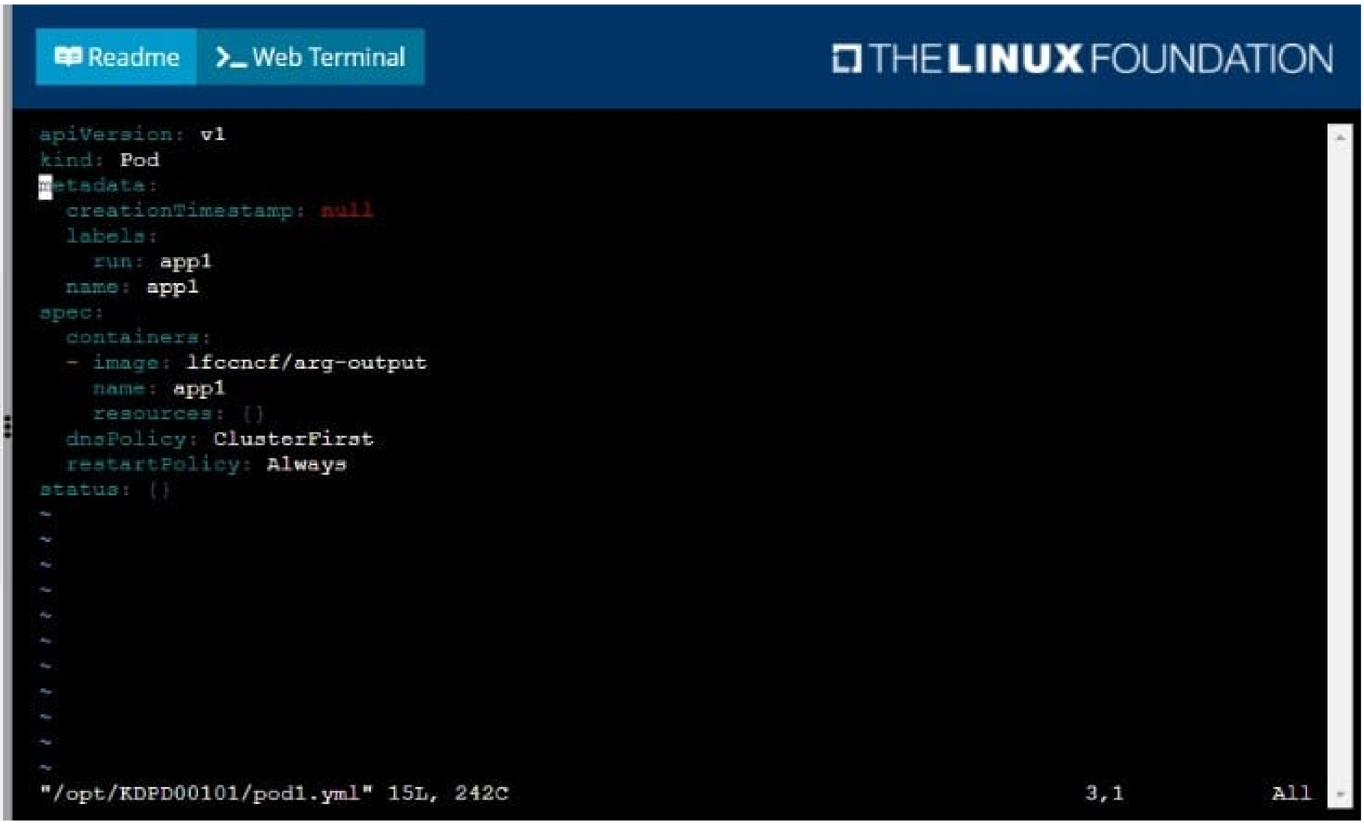
A. Please check explanations

B. Place Holder

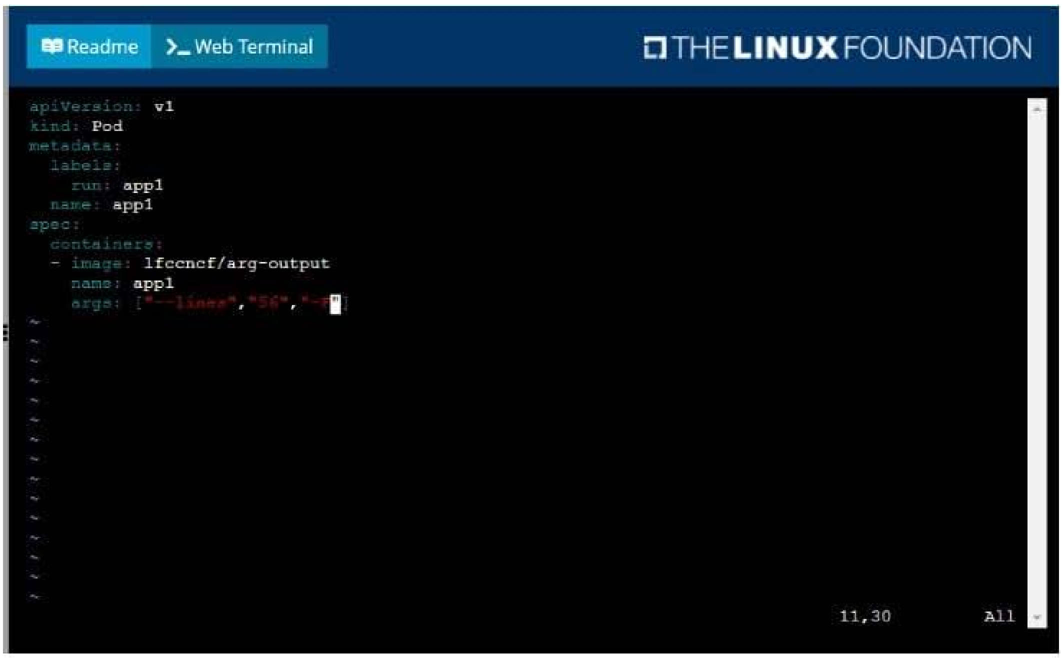
Correct Answer: A



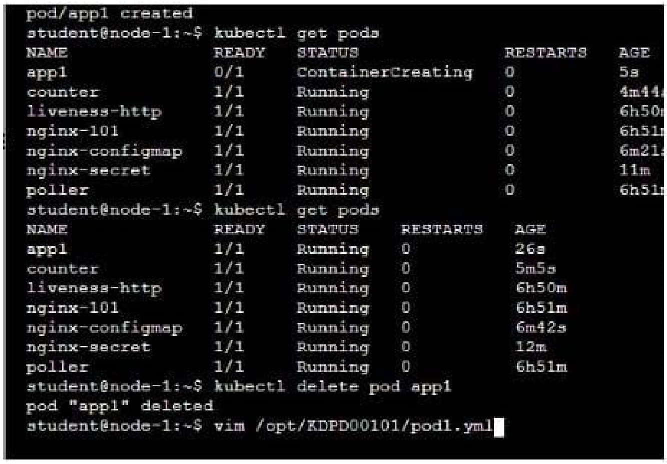
```
student@node-1:~$ kubectl run app1 --image=lfccncf/arg-output --dry-run=client -o yaml > /opt/KD
PD00101/pod1.yml
student@node-1:~$ vim /opt/KDPD00101/pod1.yml
```



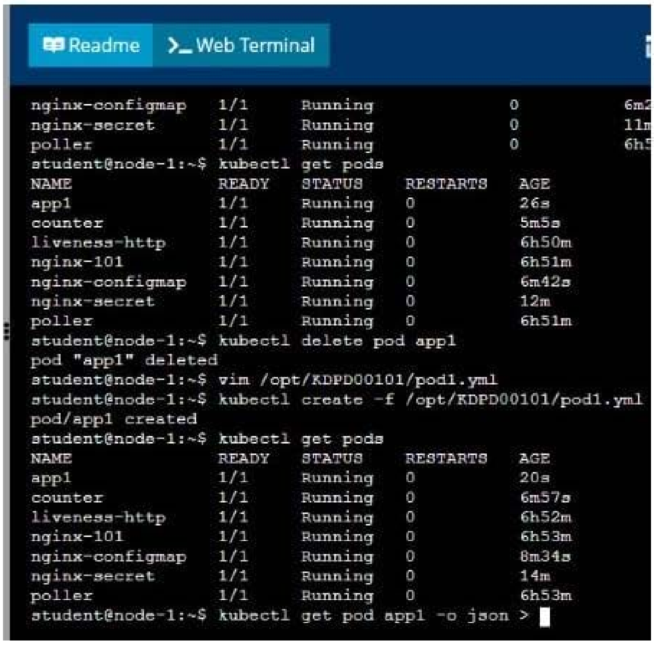
```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: app1
  name: app1
spec:
  containers:
  - image: lfccncf/arg-output
    name: app1
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
~
~
~
~
~
~
~
~
~
~
~
"/opt/KDPD00101/pod1.yml" 15L, 242C                                        3,1           All
```



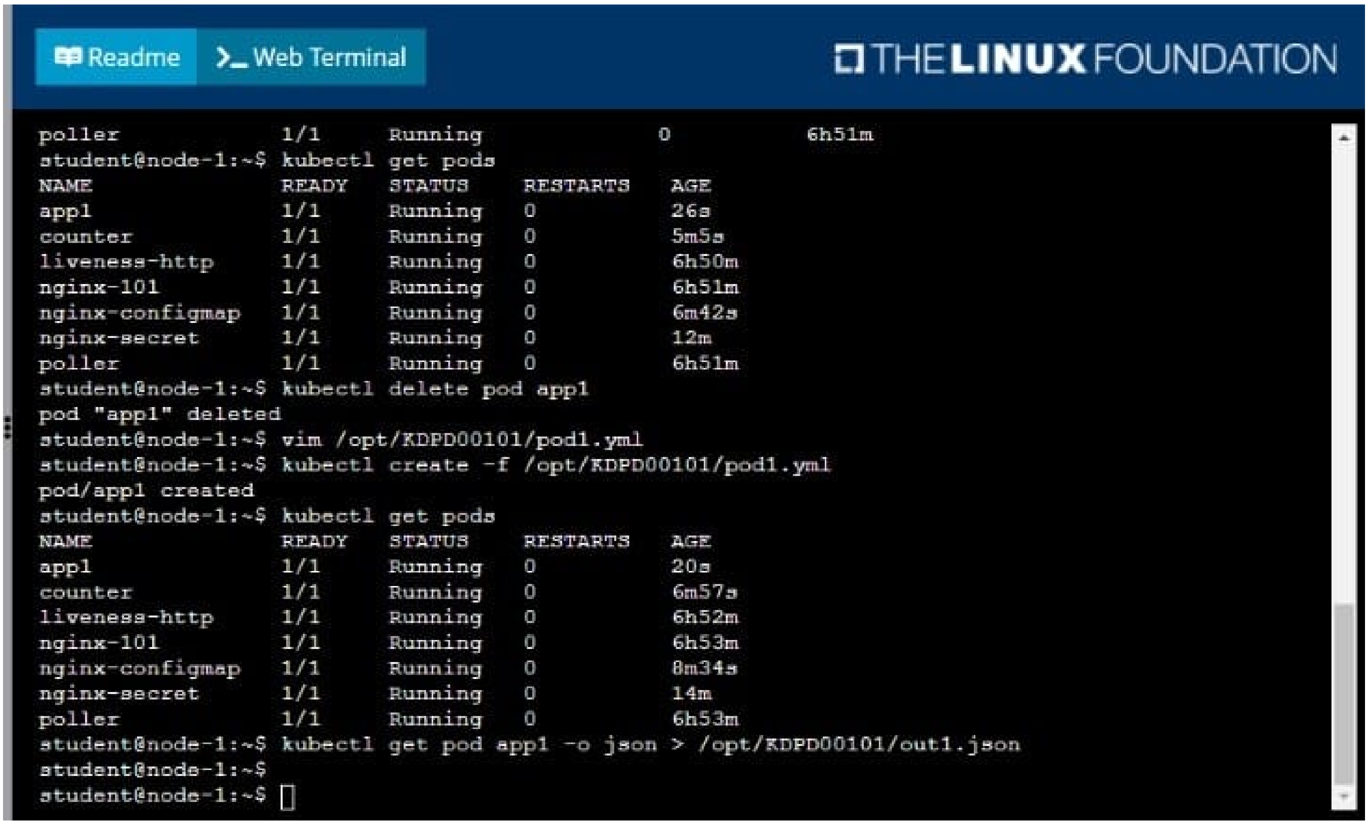
```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: app1
  name: app1
spec:
  containers:
  - image: lfccncf/arg-output
    name: app1
    args: ["--lines","56","-F"]
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                          11,30         All
```



```
pod/app1 created
student@node-1:~$ kubectl get pods
NAME             READY   STATUS              RESTARTS   AGE
app1             0/1     ContainerCreating   0          5s
counter          1/1     Running             0          4m44
liveness-http    1/1     Running             0          6h50
nginx-101        1/1     Running             0          6h51
nginx-configmap  1/1     Running             0          6m21
nginx-secret     1/1     Running             0          11m
poller           1/1     Running             0          6h51
student@node-1:~$ kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
app1             1/1     Running   0          26s
counter          1/1     Running   0          5m5s
liveness-http    1/1     Running   0          6h50m
nginx-101        1/1     Running   0          6h51m
nginx-configmap  1/1     Running   0          6m42s
nginx-secret     1/1     Running   0          12m
poller           1/1     Running   0          6h51m
student@node-1:~$ kubectl delete pod app1
pod "app1" deleted
student@node-1:~$ vim /opt/KDPD00101/pod1.yml
```



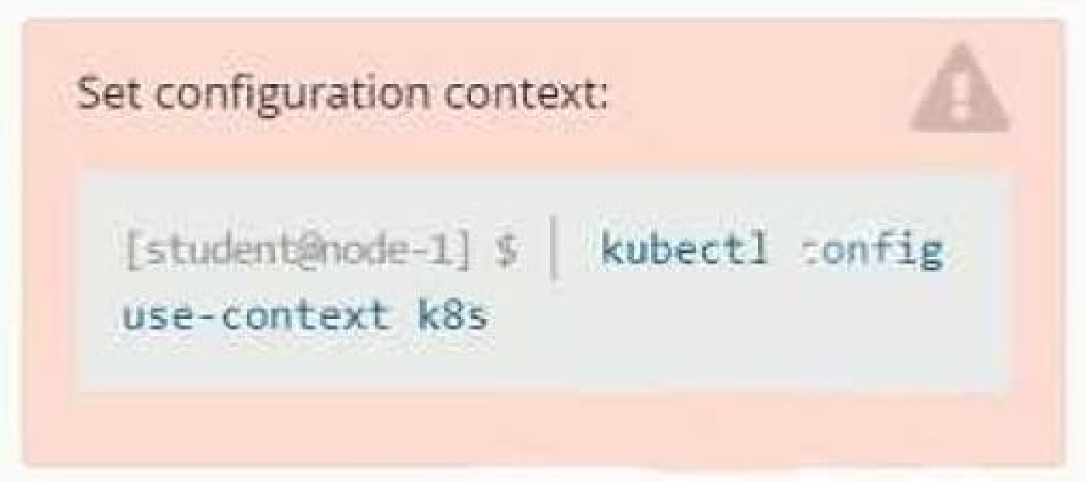
```
nginx-configmap  1/1     Running             0          6m2
nginx-secret     1/1     Running             0          11m
poller           1/1     Running             0          6h5
student@node-1:~$ kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
app1             1/1     Running   0          26s
counter          1/1     Running   0          5m5s
liveness-http    1/1     Running   0          6h50m
nginx-101        1/1     Running   0          6h51m
nginx-configmap  1/1     Running   0          6m42s
nginx-secret     1/1     Running   0          12m
poller           1/1     Running   0          6h51m
student@node-1:~$ kubectl delete pod app1
pod "app1" deleted
student@node-1:~$ vim /opt/KDPD00101/pod1.yml
student@node-1:~$ kubectl create -f /opt/KDPD00101/pod1.yml
pod/app1 created
student@node-1:~$ kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
app1             1/1     Running   0          20s
counter          1/1     Running   0          6m57s
liveness-http    1/1     Running   0          6h52m
nginx-101        1/1     Running   0          6h53m
nginx-configmap  1/1     Running   0          8m34s
nginx-secret     1/1     Running   0          14m
poller           1/1     Running   0          6h53m
student@node-1:~$ kubectl get pod app1 -o json >
```



```
poller              1/1     Running           0          6h51m
student@node-1:~$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
app1                1/1     Running   0          26s
counter             1/1     Running   0          5m5s
liveness-http       1/1     Running   0          6h50m
nginx-101           1/1     Running   0          6h51m
nginx-configmap     1/1     Running   0          6m42s
nginx-secret        1/1     Running   0          12m
poller              1/1     Running   0          6h51m
student@node-1:~$ kubectl delete pod app1
pod "app1" deleted
student@node-1:~$ vim /opt/KDPD00101/pod1.yml
student@node-1:~$ kubectl create -f /opt/KDPD00101/pod1.yml
pod/app1 created
student@node-1:~$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
app1                1/1     Running   0          20s
counter             1/1     Running   0          6m57s
liveness-http       1/1     Running   0          6h52m
nginx-101           1/1     Running   0          6h53m
nginx-configmap     1/1     Running   0          8m34s
nginx-secret        1/1     Running   0          14m
poller              1/1     Running   0          6h53m
student@node-1:~$ kubectl get pod app1 -o json > /opt/KDPD00101/out1.json
student@node-1:~$
student@node-1:~$
```

QUESTION 5

CORRECT TEXT



Set configuration context:

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

Context

A pod is running on the cluster but it is not responding.

Task

The desired behavior is to have Kubemetes restart the pod when an endpoint returns an HTTP 500 on the /healthz endpoint. The service, probe-pod, should never send traffic to the pod while it is failing. Please complete the following:

1.

The application has an endpoint, /started, that will indicate if it can accept traffic by returning an HTTP 200. If the endpoint returns an HTTP 500, the application has not yet finished initialization.

2.

The application has another endpoint /healthz that will indicate if the application is still working as expected by returning an HTTP 200. If the endpoint returns an HTTP 500 the application is no longer responsive.

3.

Configure the probe-pod pod provided to use these endpoints

4.

The probes should use port 8080

A. Please check explanations

B. Place Holder

Correct Answer: A

```
apiVersion: v1
kind: Pod
metadata:
labels:
test: liveness
name: liveness-exec
spec:
containers:
-name: liveness
image: k8s.gcr.io/busybox
args:
-/bin/sh
- -c
-touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600 livenessProbe:
exec:
command:
-cat
```

```
-/tmp/healthy

initialDelaySeconds: 5

periodSeconds: 5
```

In the configuration file, you can see that the Pod has a single Container. The periodSeconds field specifies that the kubelet should perform a liveness probe every 5 seconds. The initialDelaySeconds field tells the kubelet that it should wait 5

seconds before performing the first probe. To perform a probe, the kubelet executes the command cat /tmp/healthy in the target container. If the command succeeds, it returns 0, and the kubelet considers the container to be alive and healthy.

If the command returns a non-zero value, the kubelet kills the container and restarts it.

When the container starts, it executes this command:

/bin/sh -c "touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600" For the first 30 seconds of the container\\'s life, there is a /tmp/healthy file. So during the first 30 seconds, the command cat /tmp/healthy returns a success code. After 30

seconds, cat /tmp/healthy returns a failure code.

Create the Pod:

kubectl apply -f https://k8s.io/examples/pods/probe/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 "k8s.gcr.io/busybox"

23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image "k8s.gcr.io/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 "k8s.gcr.io/busybox"

36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image "k8s.gcr.io/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 1m