

## CKAD Dumps

### Certified Kubernetes Application Developer (CKAD) Program

<https://www.certleader.com/CKAD-dumps.html>



**NEW QUESTION 1**

Exhibit:



Context

It is always useful to look at the resources your applications are consuming in a cluster. Task

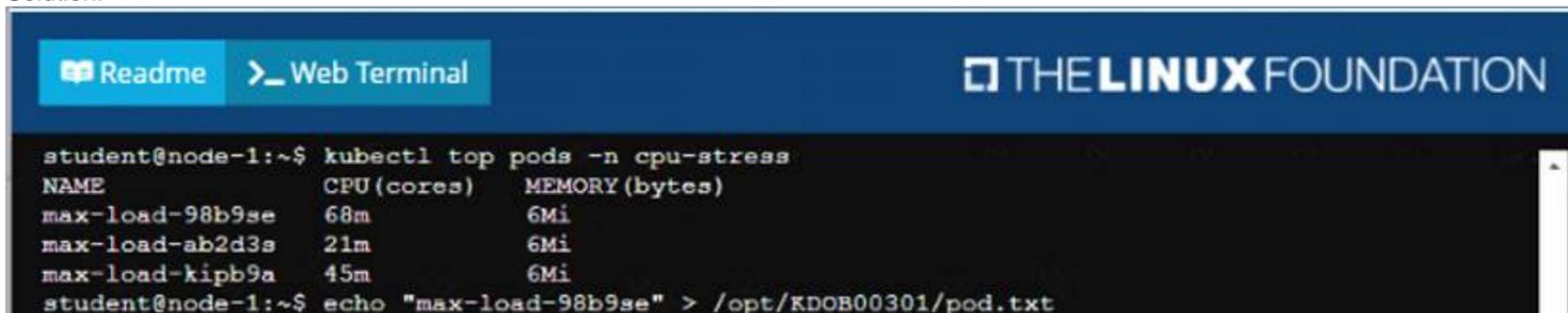
- From the pods running in namespacecpu-stress , write the name only of the pod that is consuming the most CPU to file /opt/KDOBG0301/pod.txt, which has already been created.

- A. Mastered
- B. Not Mastered

**Answer: A**

**Explanation:**

Solution:



**NEW QUESTION 2**

Exhibit:



Context

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

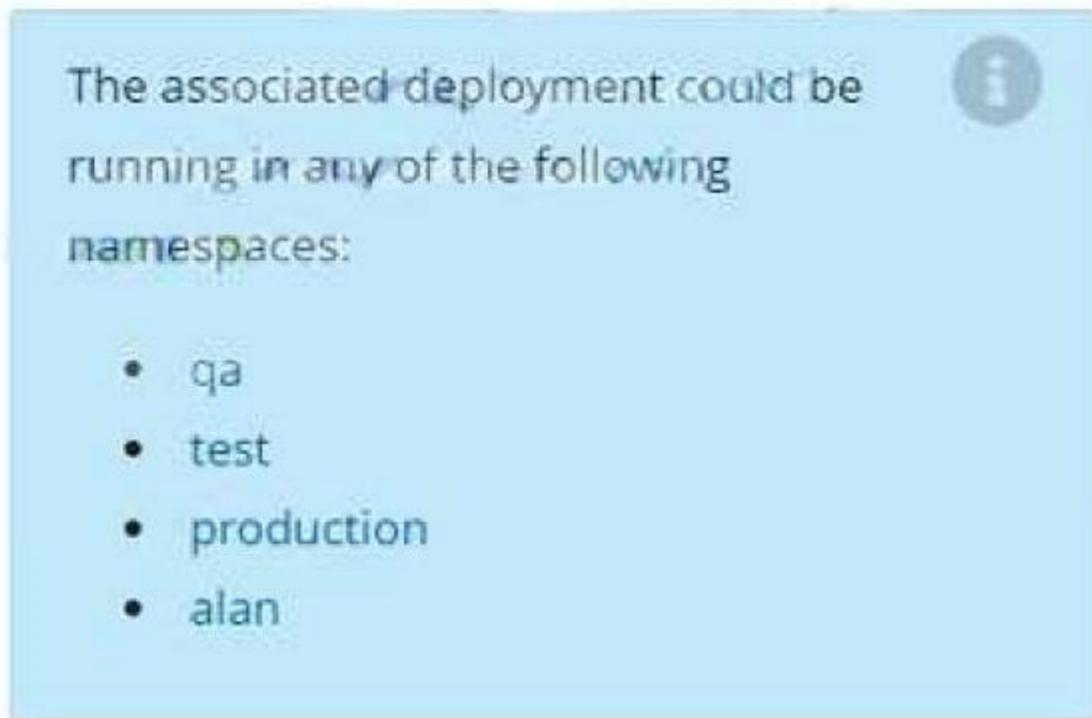
Perform the following tasks:

- Find the broken pod and store its name and namespace to /opt/KDOBG00401/broken.txt in the format:



The output file has already been created

- Store the associated error events to a file /opt/KDOBG00401/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.



- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Solution:

Create the Pod: `kubectl create -f`

<http://k8s.io/docs/tasks/configure-pod-container/execute-liveness-probe>

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 id86849c15382e; Security:[seccomp=unconfined]
23s 23s 1{kubelet worker0} spec.containers{liveness} Normal Started Started container with docker id86849c15382e
```

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 containerwithdocker id86849c15382e; Security:[seccomp=unconfined]
36s 36s 1{kubelet worker0} spec.containers{liveness} Normal Started Started containerwithdocker id86849c15382e
2s 2s 1{kubelet worker0} spec.containers{liveness} Warning Unhealthy Liveness probe failed: cat: can't open
'/tmp/healthy': No suchfileordirectory
```

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:

NAMEREADY STATUSRESTARTS AGE

liveness-exec 1/1Running 1m

**NEW QUESTION 3**

Exhibit:



Context

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

The desired behavior is to have Kubernetes 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:

- 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.
- 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.
- Configure the probe-pod pod provided to use these endpoints

- The probes should use port 8080

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Solution:

```
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;sleep30;rm-rf/tmp/healthy;sleep600
```

```
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
```

**NEW QUESTION 4**

Exhibit:



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
```

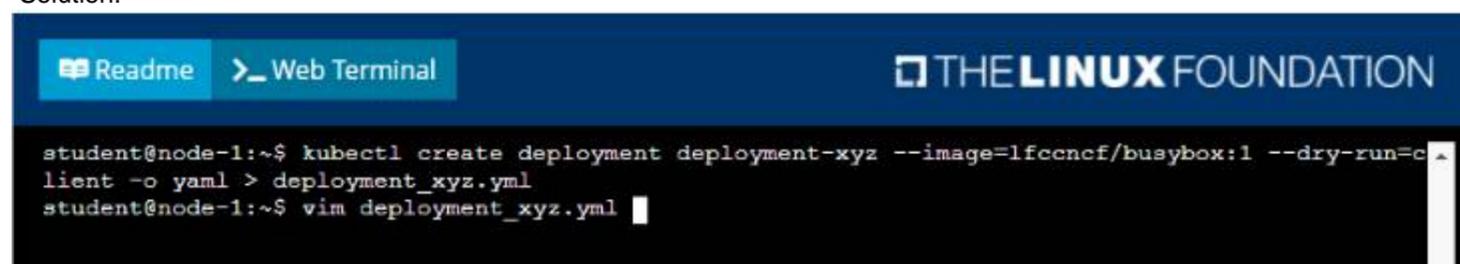
• The adapter-zen sidecar container should read /tmp/log/input.log and output the data to /tmp/log/output.\* in Fluentd JSON format. Note that no knowledge of Fluentd is required to complete this task: all you will need to achieve this is to create the ConfigMap from the spec file provided at /opt/KDMC00102/fluentd-configmap.p.yaml,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

- A. Mastered
- B. Not Mastered

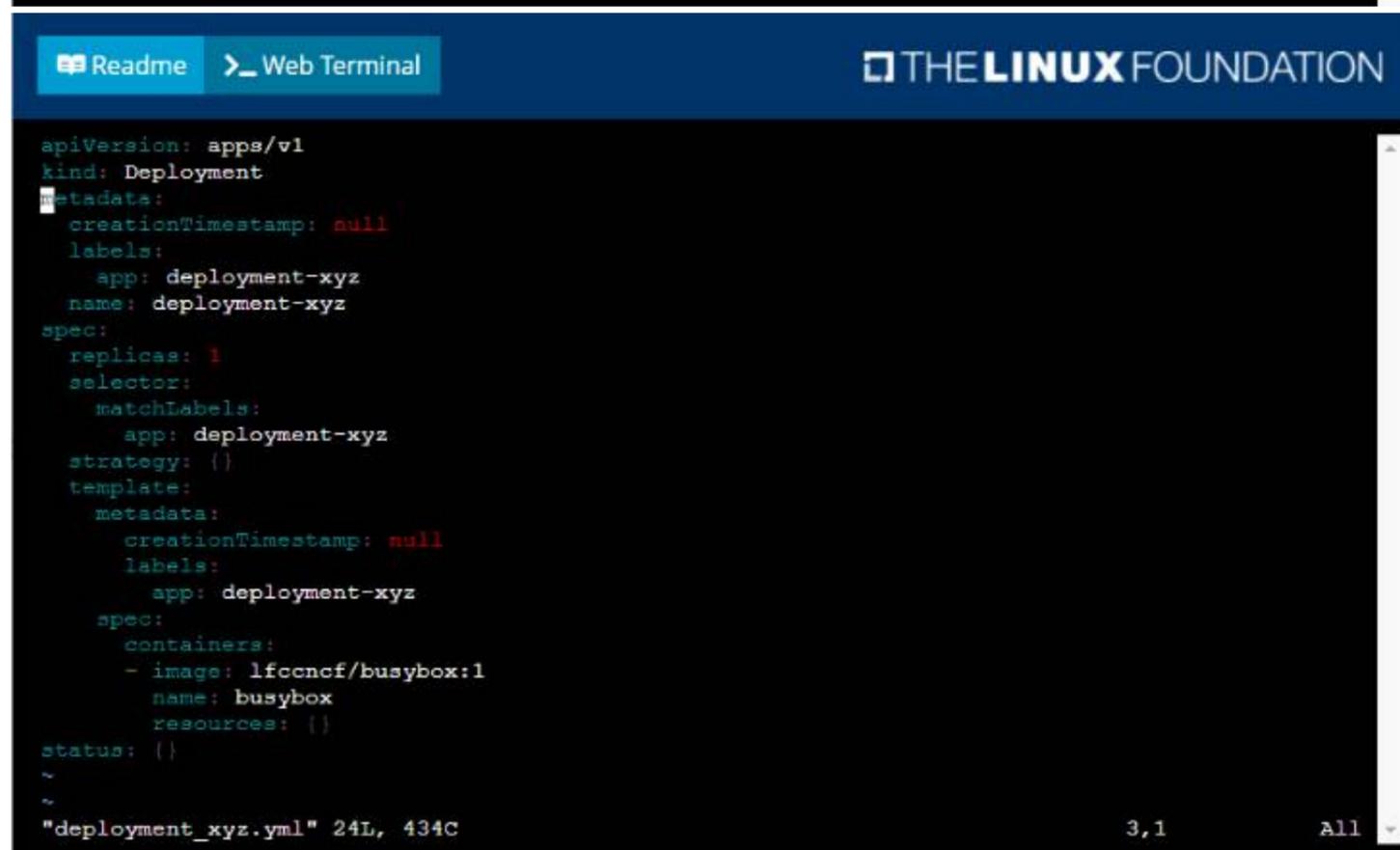
**Answer:** A

**Explanation:**

Solution:



```
student@node-1:~$ kubectl create deployment deployment-xyz --image=lfcncf/busybox:1 --dry-run=client -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 3,1 All
```

Readme Web Terminal THE LINUX FOUNDATION

```

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

```

Readme Web Terminal THE LINUX FOUNDATION

```

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
    sep 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/et
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:~$

```

```

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:~$

```

**NEW QUESTION 5**

Exhibit:



Context

Your application's namespace requires a specific service account to be used.

Task

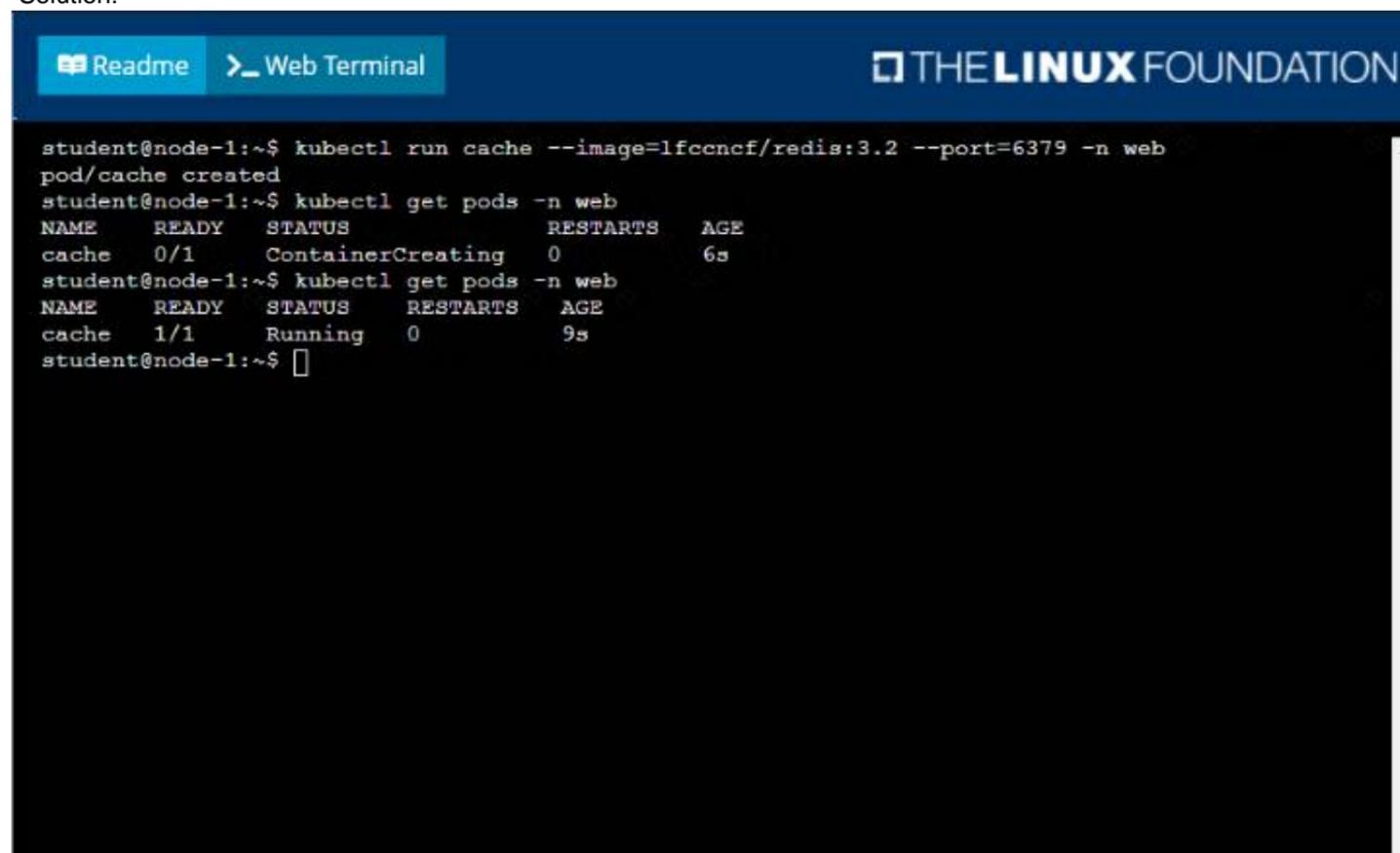
Update the app-deployment in the production namespace to run as the restricted-service-account. The service account has already been created.

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Solution:



**NEW QUESTION 6**

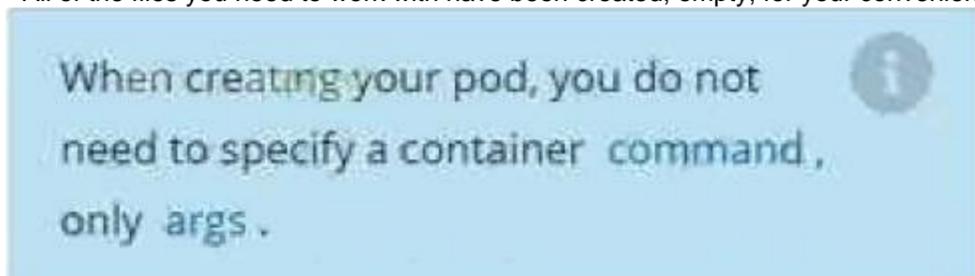
Context

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:

- Create a YAML formatted pod manifest /opt/KDPD00101/pod.yml to create a pod named app1 that runs a container named app1cont using image lfcncf/arg-output with these command line arguments: -lines 56 -F
- Create the pod with the kubectl command using the YAML file created in the previous step
- When the pod is running display summary data about the pod in JSON format using the kubectl command and redirect the output to a file named /opt/KDPD00101/out1.json
- All of the files you need to work with have been created, empty, for your convenience



- A. Mastered
- B. Not Mastered

**Answer:** A



```

Readme Web Terminal
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 >

```

```

Readme Web Terminal THE LINUX FOUNDATION
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:~$

```

**NEW QUESTION 7**

Exhibit:



Context

As a Kubernetes application developer you will often find yourself needing to update a running application. Task Please complete the following:

- Update the app deployment in the kdpd00202 namespace with a maxSurge of 5% and a maxUnavailable of 2%
- Perform a rolling update of the web1 deployment, changing the lfcncf/ngmx image version to 1.13
- Roll back the app deployment to the previous version

- A. Mastered
- B. Not Mastered

**Answer: A**

**Explanation:**

Solution:

```

Readme Web Terminal THE LINUX FOUNDATION
student@node-1:~$ kubectl edit deployment app -n kdpd00202

```

```

Readme Web Terminal THE LINUX FOUNDATION
uid: 1dfa2527-5c61-46a9-8dd3-e24643d3ce14
spec:
  progressDeadlineSeconds: 600
  replicas: 10
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    rollingUpdate:
      maxSurge: 5%
      maxUnavailable: 2
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
    labels:
      app: nginx
    spec:
      containers:
      - image: lfocncf/nginx:1.13
        imagePullPolicy: IfNotPresent
        name: nginx
        ports:
        - containerPort: 80
          protocol: TCP
:wq!

```

```

Readme Web Terminal THE LINUX FOUNDATION
student@node-1:~$ kubectl edit deployment app -n kdpd00202
deployment.apps/app edited
student@node-1:~$ kubectl rollout status deployment app -n kdpd00202
Waiting for deployment "app" rollout to finish: 6 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 7 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 7 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 7 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 8 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 8 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 8 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 8 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 9 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 9 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 9 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "app" rollout to finish: 8 of 10 updated replicas are available...
Waiting for deployment "app" rollout to finish: 9 of 10 updated replicas are available...
deployment "app" successfully rolled out
student@node-1:~$ kubectl rollout undo deployment app -n kdpd00202
deployment.apps/app rolled back
student@node-1:~$ kubectl rollout status deployment app -n kdpd00202

```

```

student@node-1:~$ kubectl rollout status deployment app -n kdpd00202
Waiting for deployment "app" rollout to finish: 6 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 6 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 6 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 6 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 7 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 9 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 9 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 9 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "app" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "app" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "app" rollout to finish: 8 of 10 updated replicas are available...
Waiting for deployment "app" rollout to finish: 9 of 10 updated replicas are available...
deployment "app" successfully rolled out
student@node-1:~$

```

**NEW QUESTION 10**

.....

## Thank You for Trying Our Product

\* 100% Pass or Money Back

All our products come with a 90-day Money Back Guarantee.

\* One year free update

You can enjoy free update one year. 24x7 online support.

\* Trusted by Millions

We currently serve more than 30,000,000 customers.

\* Shop Securely

All transactions are protected by VeriSign!

**100% Pass Your CKAD Exam with Our Prep Materials Via below:**

<https://www.certleader.com/CKAD-dumps.html>