



Linux-Foundation

Exam Questions CKAD

Certified Kubernetes Application Developer (CKAD) Program



About Exambible

Your Partner of IT Exam

Found in 1998

Exambible is a company specialized on providing high quality IT exam practice study materials, especially Cisco CCNA, CCDA, CCNP, CCIE, Checkpoint CCSE, CompTIA A+, Network+ certification practice exams and so on. We guarantee that the candidates will not only pass any IT exam at the first attempt but also get profound understanding about the certificates they have got. There are so many alike companies in this industry, however, Exambible has its unique advantages that other companies could not achieve.

Our Advances

* 99.9% Uptime

All examinations will be up to date.

* 24/7 Quality Support

We will provide service round the clock.

* 100% Pass Rate

Our guarantee that you will pass the exam.

* Unique Gurantee

If you do not pass the exam at the first time, we will not only arrange FULL REFUND for you, but also provide you another exam of your claim, ABSOLUTELY FREE!



NEW QUESTION 1

Exhibit:



Context

A user has reported an aopticauon 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.

The associated deployment could be running in any of the following namespaces:

- ds
- test
- production
- alan

A. Mastered

B. Not Mastered

Answer: A

Explanation:

Solution:

Create the Pod: kubectlcreate-f

http://k8s.io/docs/tasks/configure-pod-container/

exec-liveness.yaml

Within 30 seconds, view the Pod events: kubectldescribepod liveness-exec

The output indicates that no liveness probes have failed yet:

FirstSeen LastSeen CountFrom SubobjectPath Type Reason Message

24s 24s 1{default-scheduler} NormalScheduled Successfully assigned liveness-exec to worker0

23s 23s 1{kubelet worker0} spec.containers{liveness} NormalPulling pulling image"gcr.io/google_containers/busybox"

23s 23s 1{kubelet worker0} spec.containers{liveness} NormalPulled Successfully pulled image "gcr.io/google_containers/busybox"

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

 $23s\ 23s\ 1 \\ \{kubelet\ worker0\}\ spec. containers \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ container\ with\ docker\ id 86849c 15382e\ spec. \\ \{liveness\}\ Normal Started\ Started\ spec. \\ \{liveness\}\ Normal Started\ spec. \\ \{liveness\}\ Normal$

After 35 seconds, view the Pod events again: kubectldescribepod 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-exectoworker0

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 2 Exhibit:



Task

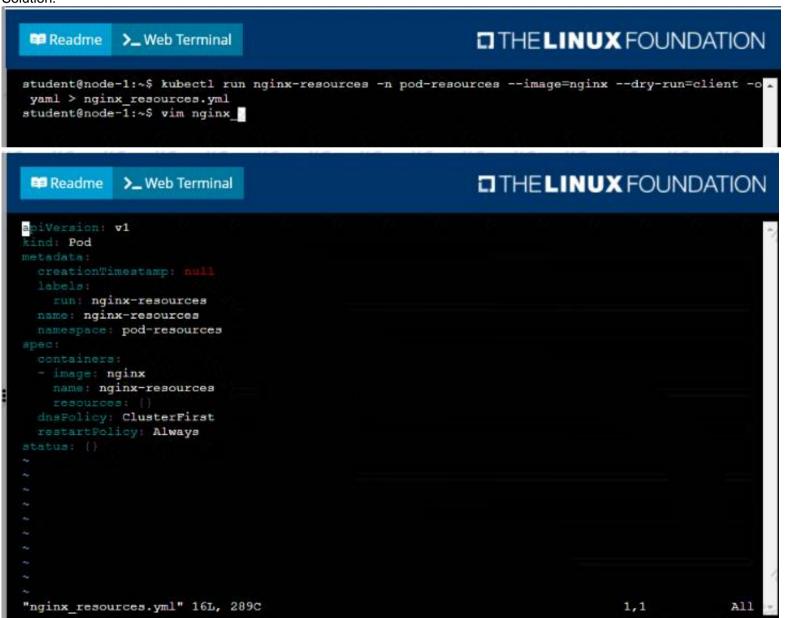
You are required to create a pod that requests a certain amount of CPU and memory, so it gets scheduled to-a node that has those resources available.

- Create a pod named nginx-resources in the pod-resources namespace that requests a minimum of 200m CPU and 1Gi memory for its container
- The pod should use the nginx image
- The pod-resources namespace has already been created
- A. Mastered
- B. Not Mastered

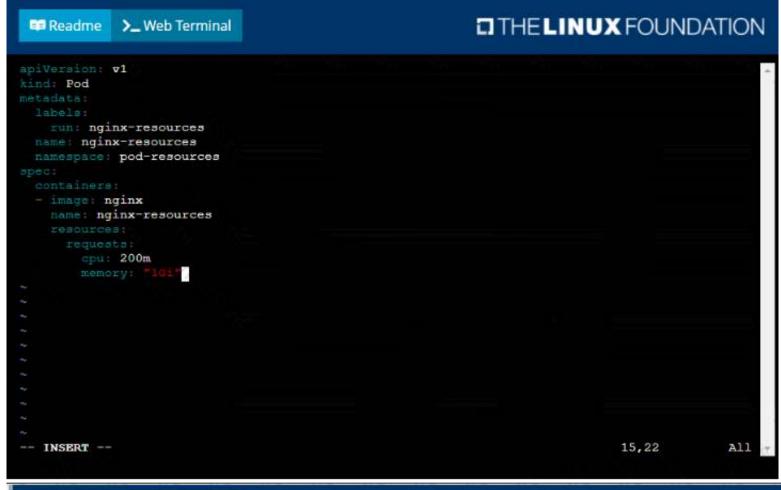
Answer: A

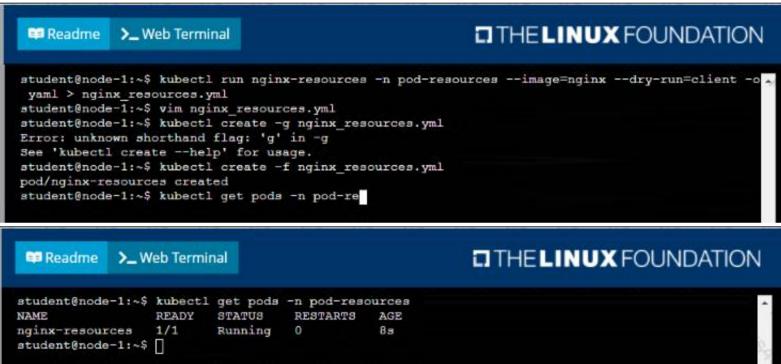
Explanation:

Solution:









NEW QUESTION 3

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

Ifccncf/busybox:1 container, named logger-dev

- •includes a sidecar lfccncf/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/fluentdconfigma p.yaml, and mount that ConfigMap to /fluentd/etc in the adapter-zen sidecar container
- A. Mastered
- B. Not Mastered

Answer: A

```
Explanation:
Solution:
                                                       THE LINUX FOUNDATION
  Readme
             >_ Web Terminal
 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
                                                       THE LINUX FOUNDATION
  Readme
             >_ Web Terminal
 apiversion: apps/vl
  kind: Deployment
 metadata:
     app: deployment-xyz
   name: deployment-xyz
       app: deployment-xyz
        app: deployment-xyz
       - image: lfccncf/busybox:1
        name: busybox
                                                                                     All
  "deployment_xyz.yml" 24L, 434C
                                                                         3,1
                                                        THE LINUX FOUNDATION
   Readme >_ Web Terminal
  kind: Deployment
```

app: deployment-xyz name: deployment-xyz deployment-xyz app: deployment-xyz - name: myvol1 - image: lfccncf/busybox:1
name: logger-dev - name: myvol1 mountPath: /tmp/log - image: lfccncf/fluentd:v0.12 name: adapter-zen 27,22 3 lines yanked Bot

Your Partner of IT Exam visit - https://www.exambible.com



```
THE LINUX FOUNDATION
Readme >_ Web Terminal
      app: deployment-xyz
    - name: myvol1
    - name: myvol2
       name: logconf
    - image: lfccncf/busybox:1
      name: logger-dev
      - name: myvol1
       mountPath: /tmp/log
    - image: lfccncf/fluentd:v0.12
      name: adapter-zen
      command: ["/bin
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
                READY UP-TO-DATE
                                   AVAILABLE
                                               AGE
deployment-xyz 0/1
                                    0
                                               55
student@node-1:~$ kubectl get deployment
               READY UP-TO-DATE AVAILABLE
deployment-xyz 0/1
                                    0
                       1
student@node-1:~$ kubectl get deployment
               READY UP-TO-DATE AVAILABLE
                                               AGE
deployment-xyz 1/1
                                    1
                                               125
student@node-1:~$
student@node-1:~$ kubectl create -f deployment_xyz.yml
deployment.apps/deployment-xyz created
student@node-1:~$ kubectl get deployment
               READY UP-TO-DATE AVAILABLE
deployment-xyz 0/1
                       1
                                    0
student@node-1:~$ kubectl get deployment
               READY
                       UP-TO-DATE
                                   AVAILABLE
                                               AGE
deployment-xyz 0/1
student@node-1:~$ kubectl get deployment
               READY UP-TO-DATE AVAILABLE
deployment-xyz 1/1
                       1
                                    1
                                               125
student@node-1:~$
```

NEW QUESTION 4

Exhibit:



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.

Your Partner of IT Exam visit - https://www.exambible.com



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.

A. Mastered B. Not Mastered

Answer: A

Explanation:

Suggest the Solution.

NEW QUESTION 10

•••••



Relate Links

100% Pass Your CKAD Exam with Exambible Prep Materials

https://www.exambible.com/CKAD-exam/

Contact us

We are proud of our high-quality customer service, which serves you around the clock 24/7.

Viste - https://www.exambible.com/