

DUMPSQUEEN

Certified Kubernetes Application Developer (CKAD) Program

Linux Foundation CKAD

Version Demo

Total Demo Questions: 5

Total Premium Questions: 33

Buy Premium PDF

<https://dumpsqueen.com>

support@dumpsqueen.com

dumpsqueen.com

QUESTION NO: 1 - (SIMULATION)



Context

You are tasked to create a secret and consume the secret in a pod using environment variables as follow:

Task

- Create a secret named another-secret with a key/value pair; key1/value4
- Start an nginx pod named nginx-secret using container image nginx, and add an environment variable exposing the value of the secret key key 1, using COOL_VARIABLE as the name for the environment variable inside the pod

ANSWER: Seethesolutionbelow.

Explanation:

Solution:

```
student@node-1:~$ kubectl create secret generic some-secret --from-literal=key1=value4
secret/some-secret created
student@node-1:~$ kubectl get secret
NAME                                TYPE                                DATA  AGE
default-token-4kvr5                 kubernetes.io/service-account-token  3      2d11h
some-secret                          opaque                              1      5s
student@node-1:~$ kubectl run nginx-secret --image=nginx --dry-run=client -o yaml > nginx_secret
.yml
student@node-1:~$ vim nginx_secret.yml
```

```
Readme Web Terminal THE LINUX FOUNDATION
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx-secret
  name: nginx-secret
spec:
  containers:
  - image: nginx
    name: nginx-secret
  resources:
    dnsPolicy: ClusterFirst
    restartPolicy: Always
  status: {}

"nginx_secret.yml" 15L, 253C 1,1 All
```

```
Readme Web Terminal THE LINUX FOUNDATION
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx-secret
  name: nginx-secret
spec:
  containers:
  - image: nginx
    name: nginx-secret
    env:
    - name: COOL_VARIABLE
      valueFrom:
        secretKeyRef:
          name: some-secret
          key: key1
  status: {}

-- INSERT -- 16,20 All
```

```
Readme Web Terminal THE LINUX FOUNDATION
student@node-1:~$ kubectl get pods -n web
NAME      READY   STATUS    RESTARTS   AGE
cache     1/1     Running   0           9s
student@node-1:~$ kubectl create secret generic some-secret --from-literal=key=value4
secret/some-secret created
student@node-1:~$ kubectl get secret
NAME      TYPE      DATA   AGE
default-token-4kvr3   kubernetes.io/service-account-token   3       2d11h
some-secret           Opaque                                 1       5s
student@node-1:~$ kubectl run nginx-secret --image=nginx --dry-run=client -o yaml --nginx_secret
-yaml
student@node-1:~$ vim nginx_secret.yaml
student@node-1:~$ kubectl create -f nginx_secret.yaml
pod/nginx-secret created
student@node-1:~$ kubectl get pods
NAME      READY   STATUS             RESTARTS   AGE
liveness-http  1/1     Running            0           6h38m
nginx-101     1/1     Running            0           6h39m
nginx-secret   0/1     ContainerCreating  0           4s
poller        1/1     Running            0           6h39m
student@node-1:~$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
liveness-http  1/1     Running   0           6h38m
nginx-101     1/1     Running   0           6h39m
nginx-secret   1/1     Running   0           8s
poller        1/1     Running   0           6h39m
student@node-1:~$
```

QUESTION NO: 2 - (SIMULATION)



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 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

ANSWER: Seethesolutionbelow.

Explanation:

Solution:

To have Kubernetes automatically restart a pod when an endpoint returns an HTTP 500 on the /healthz endpoint, you will need to configure liveness and readiness probes on the pod.

First, you will need to create a livenessProbe and a readinessProbe in the pod's definition yaml file. The livenessProbe will check the /healthz endpoint, and if it returns an HTTP 500, the pod will be restarted. The readinessProbe will check the /started endpoint, and if it returns an HTTP 500, the pod will not receive traffic.

Here's an example of how you can configure the liveness and readiness probes in the pod definition yaml file:

```
apiVersion: v1
kind: Pod
metadata:
  name: probe-pod
spec:
  containers:
  - name: probe-pod
    ports:
    - containerPort: 8080
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8080
      initialDelaySeconds: 15
      periodSeconds: 10
```

failureThreshold: 3

readinessProbe:

httpGet:

path: /started

port: 8080

initialDelaySeconds: 15

periodSeconds: 10

failureThreshold: 3

The httpGet specifies the endpoint to check and the port to use. The initialDelaySeconds is the amount of time the pod will wait before starting the probe. periodSeconds is the amount of time between each probe check, and the failureThreshold is the number of failed probes before the pod is considered unresponsive.

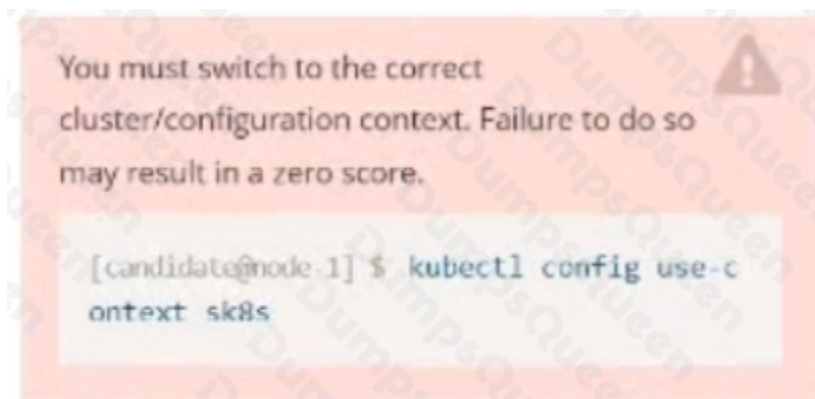
You can use kubectl to create the pod by running the following command:

Once the pod is created, Kubernetes will start monitoring it using the configured liveness and readiness probes. If the /healthz endpoint returns an HTTP 500, the pod will be restarted. If the /started endpoint returns an HTTP 500, the pod will not receive traffic.

Please note that if the failure threshold is set to 3, it means that if the probe fails 3 times consecutively it will be considered as a failure.

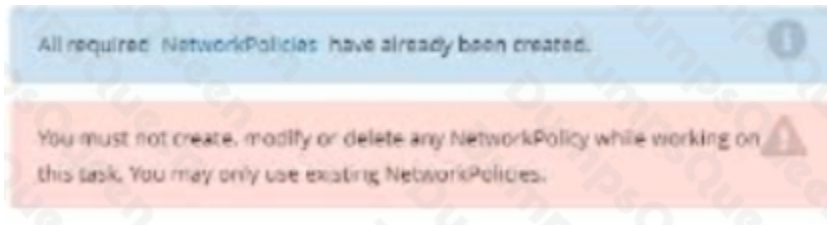
The above configuration assumes that the application is running on port 8080 and the endpoints are available on the same port.

QUESTION NO: 3 - (SIMULATION)



Task:

Update the Pod ckad00018-newpod in the ckad00018 namespace to use a NetworkPolicy allowing the Pod to send and receive traffic only to and from the pods web and db



ANSWER: Seethesolutionbelow.

Explanation:

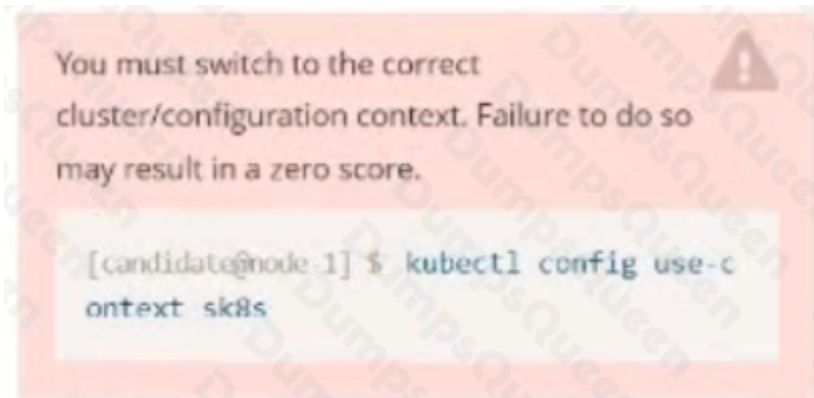
Solution:

```
candidate@node-1:~$ kubectl config use-context nk8s
Switched to context "nk8s".
candidate@node-1:~$ kubectl describe netpol -n ckad00018
```

```
File Edit View Terminal Tabs Help
name: all-access
namespace: ckad00018
created on: 2022-09-24 04:27:37 +0000 UTC
Labels: <none>
Annotations: <none>
spec:
  PodSelector: all-access=true
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From: <any> (traffic not restricted by source)
  Allowing egress traffic:
    To Port: <any> (traffic allowed to all ports)
    To: <any> (traffic not restricted by destination)
  Policy Types: Ingress, Egress

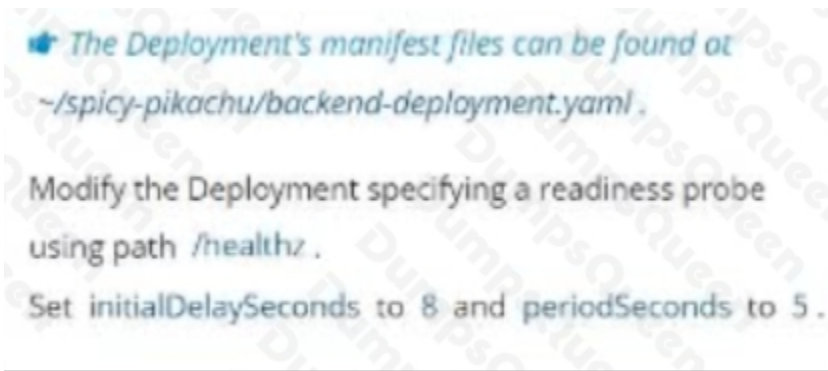
name: default-deny
namespace: ckad00018
created on: 2022-09-24 04:27:37 +0000 UTC
Labels: <none>
Annotations: <none>
spec:
  PodSelector: <none> (Allowing the specific traffic to all pods in this namespace)
  Allowing ingress traffic:
    <none> (Selected pods are isolated for ingress connectivity)
  Not affecting egress traffic
  Policy Types: Ingress
candidate@node-1:~$ kubectl label pod ckad00018-newpod -n ckad00018 web-access=true
pod/ckad00018-newpod labeled
candidate@node-1:~$ kubectl label pod ckad00018-newpod -n ckad00018 db-access=true
pod/ckad00018-newpod labeled
candidate@node-1:~$
```

QUESTION NO: 4 - (SIMULATION)



Task

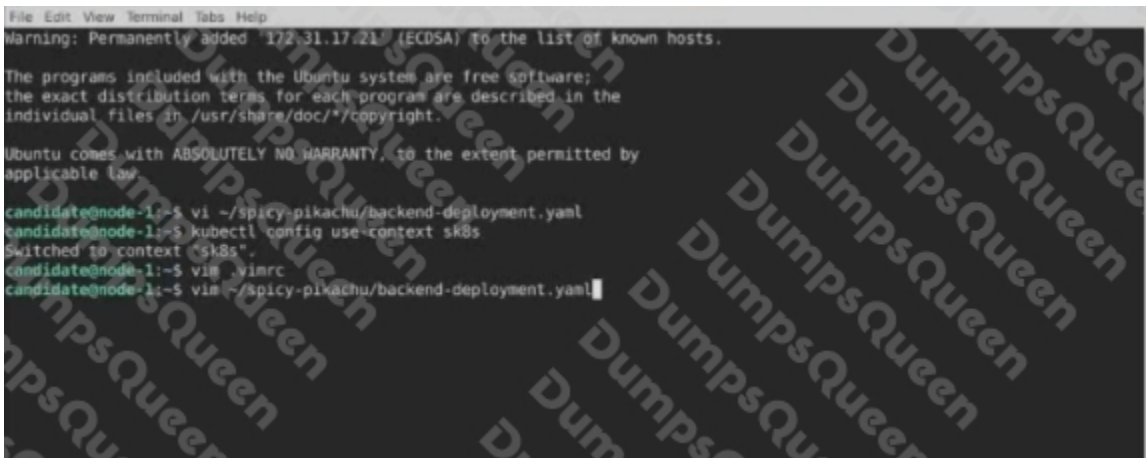
A Deployment named backend-deployment in namespace staging runs a web application on port 8081.



ANSWER: Seethesolutionbelow.

Explanation:

Solution:




```
File Edit View Terminal Tabs Help
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
  namespace: staging
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 8081
          readinessProbe:
            initialDelaySeconds: 8
            periodSeconds: 5
            httpGet:
              path: /healthz
              port: 8081
          volumeMounts:
            - mountPath: /etc/nginx/conf.d
              name: config
            - mountPath: /usr/share/nginx/html
              name: www
-- INSERT --
```

```
File Edit View Terminal Tabs Help
Warning: Permanently added '172.31.17.21' (ECDSA) to the list of known hosts.

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:~$ vi ~/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           29s
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
```

QUESTION NO: 5 - (SIMULATION)



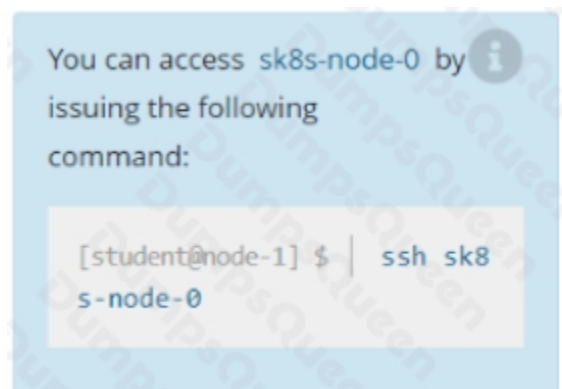
Context


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



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

ANSWER: Seethesolutionbelow.

Explanation:

Solution:

```
Readme Web Terminal THE LINUX FOUNDATION
student@node-1:~$ kubectl config use-context sk8s
Switched to context "sk8s".
student@node-1:~$
```

```
Readme Web Terminal THE LINUX FOUNDATION
* 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 eni0: 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:~$
```

```
Readme Web Terminal THE LINUX FOUNDATION
student@sk8s-node-0:~$ echo 'Acet=Finance' > /opt/KDSP00101/data/index.html
student@sk8s-node-0:~$ vim pv.yml
```

```
THE LINUX FOUNDATION
Web Terminal
INSERT 0,1 All
```

```
THE LINUX FOUNDATION
Web Terminal
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: storage
  path: /opt/K8SP00101/data
  type: Directory
```

```
THE LINUX FOUNDATION
Web Terminal
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pv-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: storage
```

```
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
```



The screenshot shows a web terminal interface with a blue header containing 'Readme', 'Web Terminal', and 'THE LINUX FOUNDATION' logo. The terminal displays a Kubernetes pod manifest for 'mypod'. The manifest includes metadata, labels, and a container named 'myfrontend' that uses a volume named 'mypod' which is backed by a PersistentVolumeClaim named 'task-pv-claim'.

```
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
```

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



The screenshot shows a web terminal interface with a blue header containing 'Readme', 'Web Terminal', and 'THE LINUX FOUNDATION' logo. The terminal displays the output of 'kubectl get pods' at three different stages: 1) 'ContainerCreating' (4s), 2) 'ContainerCreating' (8s), and 3) 'Running' (10s). The terminal also shows the user logging out and the connection closing.

```
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:~$
```