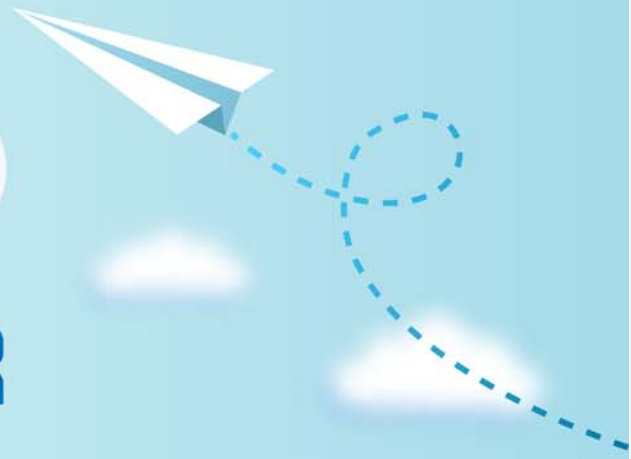


CERTPARK

QUESTION & ANSWER



CERTPARK.COM

Accurate Study Guides,
High Passing Rate!
provides update
free of charge
in one year!



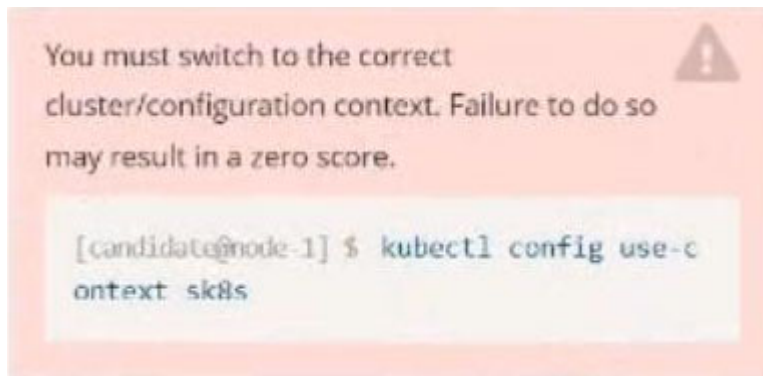
<https://www.certpark.com>

Exam : CKAD

**Title : Certified Kubernetes
Application Developer**

Version : DEMO

1.CORRECT TEXT



Task:

Create a Deployment named expose in the existing ckad00014 namespace running 6 replicas of a Pod. Specify a single container using the ifccncf/nginx: 1.13.7 image
Add an environment variable named NGINX_PORT with the value 8001 to the container then expose port 8001

Answer:

Solution:

```
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl create deploy expose -n ckad00014 --image ifccncf/nginx:1.13.7 --dry-run=client -o yaml> d
ep.yaml
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
```

```
File Edit View Terminal Tabs Help
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: expose
  name: expose
  namespace: ckad00014
spec:
  replicas: 6
  selector:
    matchLabels:
      app: expose
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: expose
    spec:
      containers:
      - image: lfccncf/nginx:1.13.7
        name: nginx
        ports:
        - containerPort: 8001
        env:
        - name: NGINX_PORT
          value: "8001"
```

Text

Description automatically generated

```

File Edit View Terminal Tabs Help
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl create deploy expose -n ckad00014 --image lfccncf/nginx:1.13.7 --dry-run=client -o yaml > d
ep.yaml
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$
candidate@node-1:~$ vim dep.yaml
candidate@node-1:~$ kubectl create -f dep.yaml
deployment.apps/expose created
candidate@node-1:~$ kubectl get pods -n ckad00014
NAME                                READY   STATUS              RESTARTS   AGE
expose-85dd99d4d9-25675             0/1     ContainerCreating   0           6s
expose-85dd99d4d9-4fhcc             0/1     ContainerCreating   0           6s
expose-85dd99d4d9-fljd7j            0/1     ContainerCreating   0           6s
expose-85dd99d4d9-tt6rm             0/1     ContainerCreating   0           6s
expose-85dd99d4d9-vjd8b             0/1     ContainerCreating   0           6s
expose-85dd99d4d9-vtzpq             0/1     ContainerCreating   0           6s
candidate@node-1:~$ kubectl get deploy -n ckad00014
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
expose    6/6     6            6           15s
candidate@node-1:~$

```

Text

Description automatically generated

2.CORRECT TEXT





Task

Create a new deployment for running nginx with the following parameters;

- Run the deployment in the kdpd00201 namespace. The namespace has already been created
- Name the deployment frontend and configure with 4 replicas
- Configure the pod with a container image of lfccncf/nginx:1.13.7
- Set an environment variable of NGINX__PORT=8080 and also expose that port for the container above



Answer:

Solution:

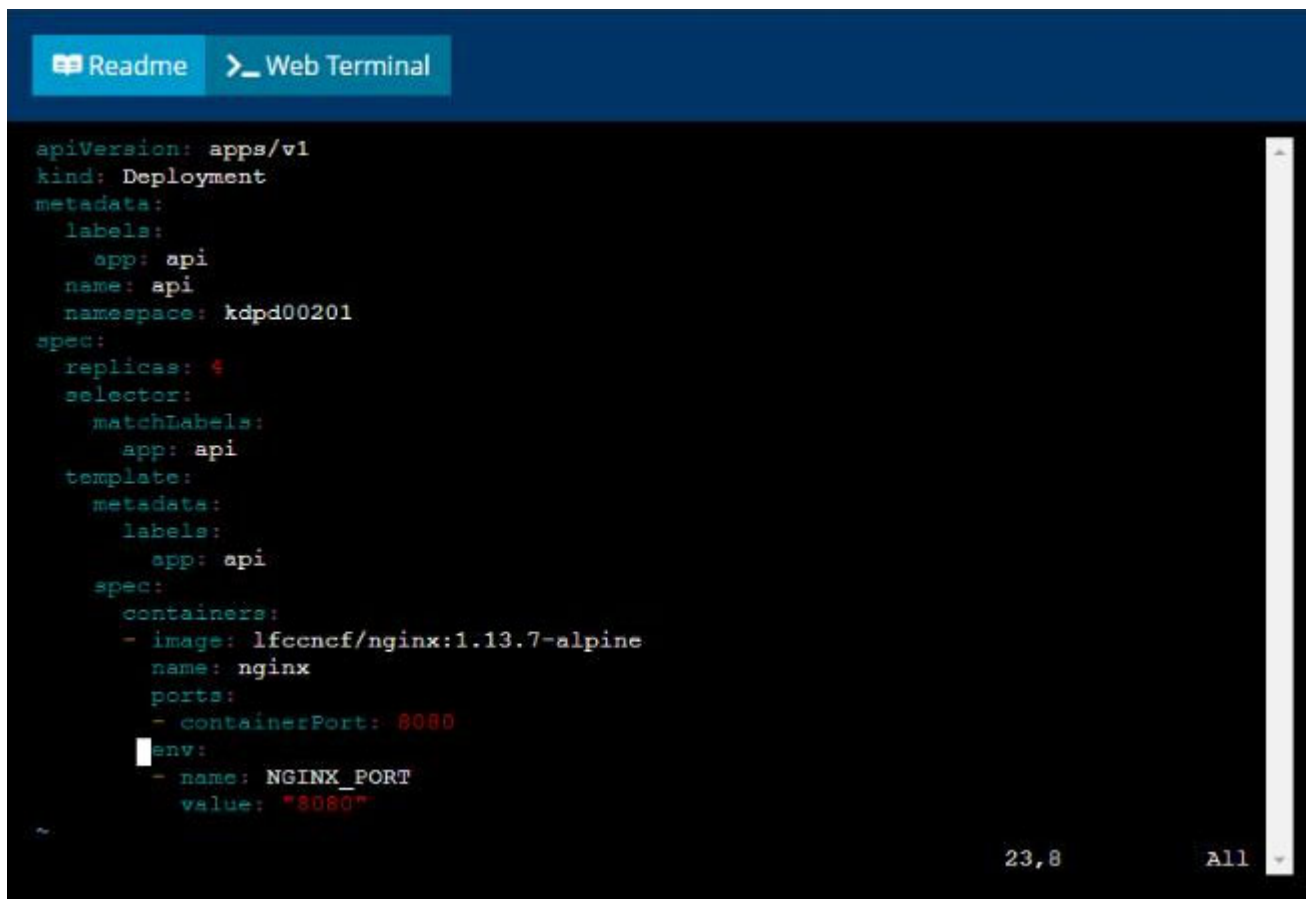
 Readme  Web Terminal

THE **LINUX** FOUNDATION

```
student@node-1:~$ kubectl create deployment api --image=lfcncf/nginx:1.13.7-alpine --replicas=4
-n kdpd00201 --dry-run=client -o yaml > nginx_deployment.yml
student@node-1:~$ vim nginx_deployment.yml
```

 Readme  Web Terminal

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: api
  name: api
  namespace: kdpd00201
spec:
  replicas: 4
  selector:
    matchLabels:
      app: api
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: api
    spec:
      containers:
      - image: lfcncf/nginx:1.13.7-alpine
        name: nginx
        resources: {}
status: {}
~
"nginx_deployment.yml" 25L, 421C 4,1 All
```

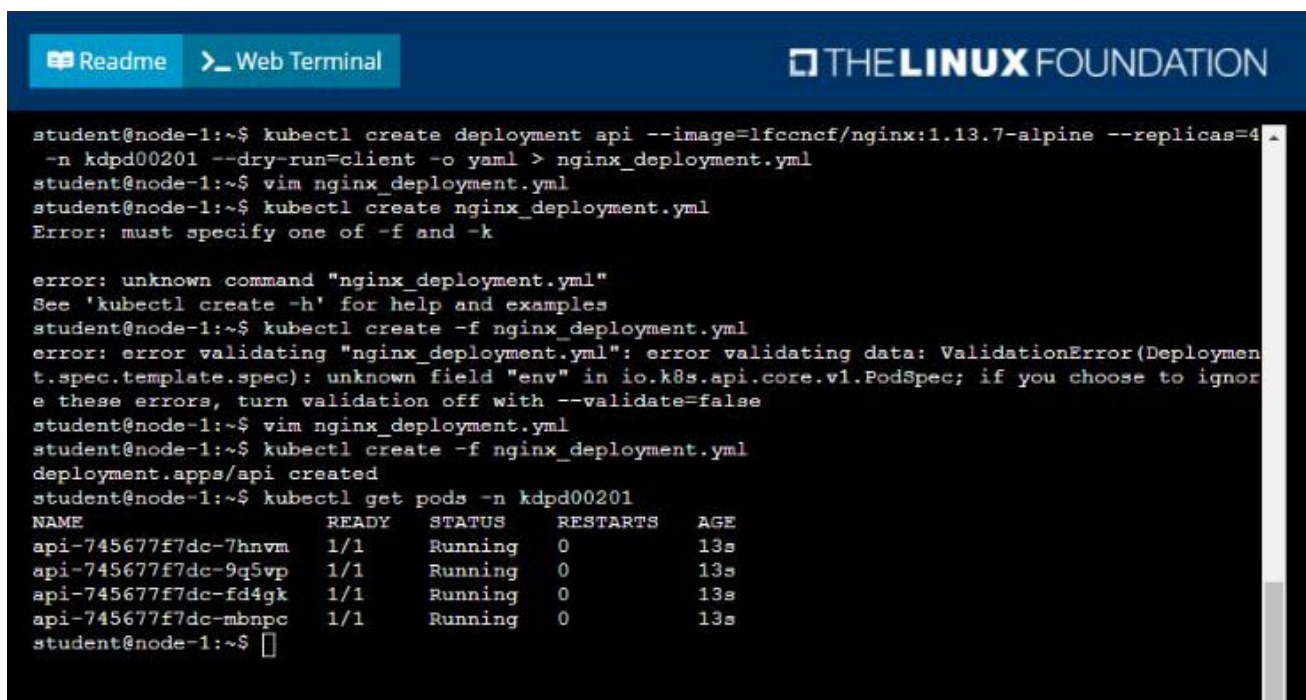


The screenshot shows a web terminal interface with a dark blue header. On the left, there are two buttons: 'Readme' and 'Web Terminal'. The main area displays a YAML configuration for a Kubernetes Deployment. The configuration includes metadata (labels: app: api, name: api, namespace: kdpg00201) and a spec (replicas: 4, selector: matchLabels: app: api, template: metadata: labels: app: api, spec: containers: - image: lfccncf/nginx:1.13.7-alpine, name: nginx, ports: - containerPort: 8080, env: - name: NGINX_PORT, value: '8080'). At the bottom right, there is a status bar showing '23,8' and 'All'.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: api
    name: api
    namespace: kdpg00201
spec:
  replicas: 4
  selector:
    matchLabels:
      app: api
  template:
    metadata:
      labels:
        app: api
    spec:
      containers:
        - image: lfccncf/nginx:1.13.7-alpine
          name: nginx
          ports:
            - containerPort: 8080
          env:
            - name: NGINX_PORT
              value: "8080"

```



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, there is a logo for 'THE LINUX FOUNDATION'. The main area displays a series of terminal commands and their outputs. The commands include creating a deployment with specific image and replicas, editing the deployment file, and then creating the deployment. The output shows the deployment being created successfully and the pods being listed in a table.

```

student@node-1:~$ kubectl create deployment api --image=lfccncf/nginx:1.13.7-alpine --replicas=4
-n kdpg00201 --dry-run=client -o yaml > nginx_deployment.yml
student@node-1:~$ vim nginx_deployment.yml
student@node-1:~$ kubectl create nginx_deployment.yml
Error: must specify one of -f and -k

error: unknown command "nginx_deployment.yml"
See 'kubectl create -h' for help and examples
student@node-1:~$ kubectl create -f nginx_deployment.yml
error: error validating "nginx_deployment.yml": error validating data: ValidationError(Deployment.spec.template.spec): unknown field "env" in io.k8s.api.core.v1.PodSpec; if you choose to ignore these errors, turn validation off with --validate=false
student@node-1:~$ vim nginx_deployment.yml
student@node-1:~$ kubectl create -f nginx_deployment.yml
deployment.apps/api created
student@node-1:~$ kubectl get pods -n kdpg00201
NAME                                READY   STATUS    RESTARTS   AGE
api-745677f7dc-7hnmv               1/1     Running   0           13s
api-745677f7dc-9q5vp               1/1     Running   0           13s
api-745677f7dc-fd4gk               1/1     Running   0           13s
api-745677f7dc-mbnpc               1/1     Running   0           13s
student@node-1:~$

```

3.CORRECT TEXT



Context

You have been tasked with scaling an existing deployment for availability, and creating a service to expose the deployment within your infrastructure.

Task

Start with the deployment named `kdsn00101-deployment` which has already been deployed to the namespace `kdsn00101` . Edit it to:

- Add the `func=webFrontEnd` key/value label to the pod template metadata to identify the pod for the service definition
- Have 4 replicas

Next, create a deployment in namespace `kdsn00101` a service that accomplishes the following:

- Exposes the service on TCP port 8080
- is mapped to the pods defined by the specification of `kdsn00101-deployment`
- Is of type `NodePort`
- Has a name of `cherry`

Answer:

Solution:

```
student@node-1:~$ kubectl edit deployment kdsn00101-deployment -n kdsn00101
```


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


```

Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
    creationTimestamp: "2020-10-09T08:50:39Z"
    generation: 1
  labels:
    app: nginx
  name: kdsn00101-deployment
  namespace: kdsn00101
  resourceVersion: "4786"
  selfLink: /apis/apps/v1/namespaces/kdsn00101/deployments/kdsn00101-deployment
  uid: 8d3ace00-7761-4189-ba10-fbc676c311bf
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
"/tmp/kubect1-edit-d4y5r.yaml" 70L, 1957C
1,1
Top

```

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

```

uid: 8d3ace00-7761-4189-ba10-fbc676c311bf
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
        func: webFrontEnd
    spec:
      containers:
      - image: nginx:latest
        imagePullPolicy: Always
        name: nginx
        ports:
        - containerPort: 80

```

```

student@node-1:~$ kubectl edit deployment kdsn00101-deployment -n kdsn00101
deployment.apps/kdsn00101-deployment edited
student@node-1:~$ kubectl get deployment kdsn00101-deployment -n kdsn00101
NAME                    READY    UP-TO-DATE    AVAILABLE    AGE
kdsn00101-deployment    4/4      4              4             7h17m
student@node-1:~$ kubectl expose deployment kdsn00101-deployment -n kdsn00101 --type NodePort --
port 8080 --name cherry
service/cherry exposed

```

4.CORRECT TEXT



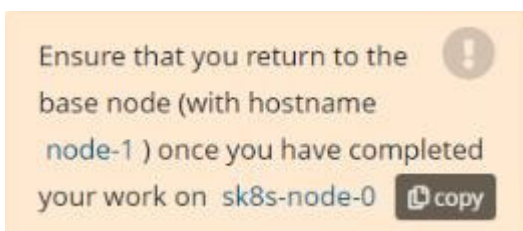
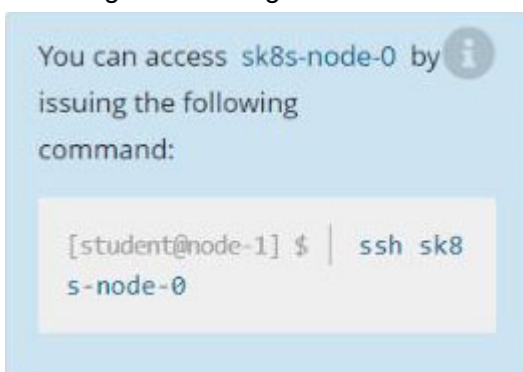
Context

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



Answer:

Solution:

```
THE LINUX FOUNDATION

Readme Web Terminal

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

```
THE LINUX FOUNDATION

Readme Web Terminal

* 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.

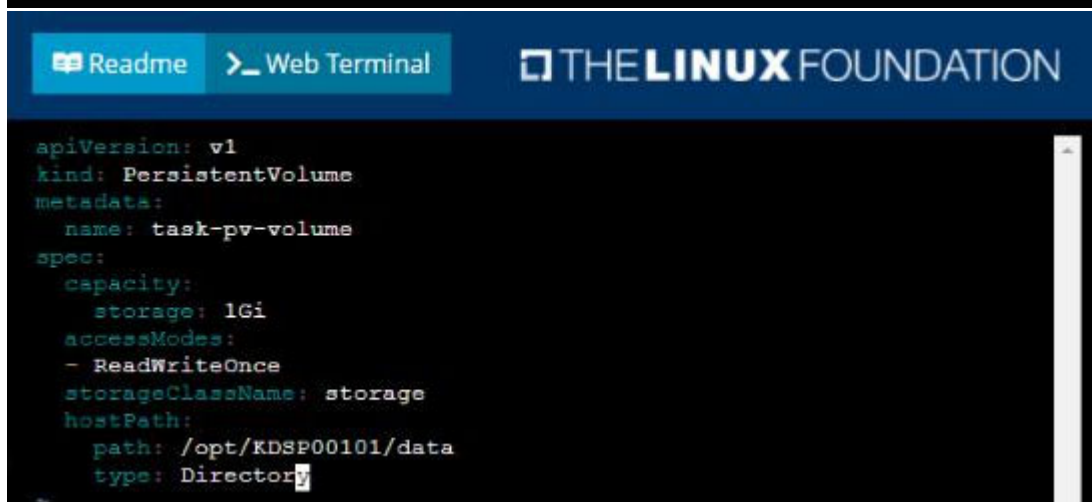
student@sk8s-node-0:~$
```

```
THE LINUX FOUNDATION

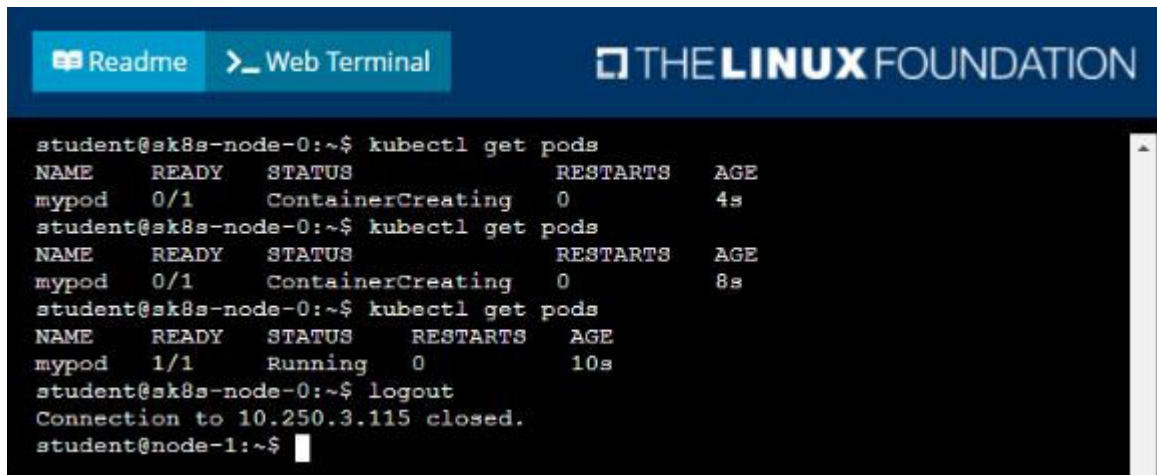
Readme Web Terminal

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

```




```
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:~$
```

5.CORRECT TEXT



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:

- 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:

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; 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