



## **Exam CKAD**

**Certified Kubernetes Application**

**Developer (CKAD) Program**

**Version: 5.0**

**[ Total Questions: 33 ]**

1. Exhibit:

```
Set configuration context:

[student@node-1] $ | kubectl config
use-context nk8s
```

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.

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.



**Answer:**

See the solution below.

Explanation:

apiVersion: networking.k8s.io/v1 kind: NetworkPolicy

metadata:

name: internal-policy namespace: default spec:

podSelector: matchLabels: name: internal policyTypes:

- Egress

- Ingress ingress:

- {}



egress:

- to:
- podSelector: matchLabels: name: mysql ports:
- protocol: TCP port: 3306
- to:
- podSelector: matchLabels:
- name: payroll ports:
- protocol: TCP port: 8080
- ports:
- port: 53 protocol: UDP
- port: 53 protocol: TCP

## 2. Exhibit:



### Context

As a Kubernetes application developer you will often find yourself needing to update a running application.

### Task

Please complete the following:

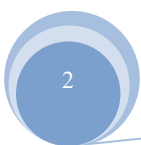
- Update the app deployment in the kdpd00202 namespace with a maxSurge of 5% and a maxUnavailable of 2%
- Perform a rolling update of the web1 deployment, changing the Ifccncf/ngmx image version to 1.13
- Roll back the app deployment to the previous version

### Answer:

See the solution below.

Explanation:

Solution:



```

THE LINUX FOUNDATION

Readme Web Terminal

student@node-1:~$ kubectl edit deployment app -n kdpd00202

```

```

THE LINUX FOUNDATION

Readme Web Terminal

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: lfccnrf/nginx:1.13
        imagePullPolicy: IfNotPresent
        name: nginx
        ports:
        - containerPort: 80
          protocol: TCP
:wg!

```

```

THE LINUX FOUNDATION

Readme Web Terminal

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: 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:~$

```



### 3. 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

#### Answer:

See the solution below.



Explanation:

Solution:

```

Readme Web Terminal THE LINUX FOUNDATION

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

```

```

Readme Web Terminal THE LINUX FOUNDATION

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

```

Readme
Web Terminal
THE **LINUX** FOUNDATION

```

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 cnf' >> /tmp/log/input.log; sl
esp 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

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 cnf' >> /tmp/log/input.log; sl
esp 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:~$

```

4. Exhibit:



```
Set configuration context:

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



Task

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

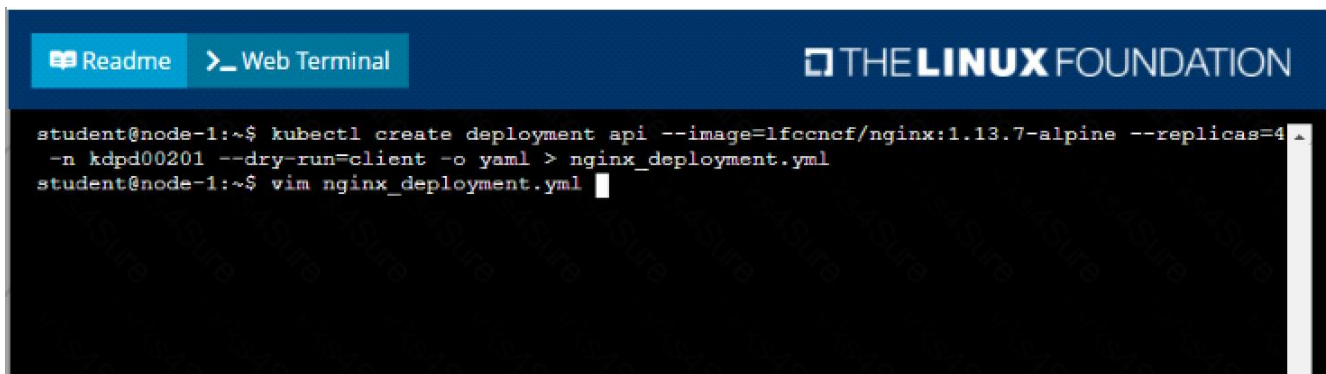
- Run the deployment in the kdpd00201 namespace. The namespace has already been created
- Name the deployment frontend and configure with 4 replicas
- Configure the pod with a container image of lfccncf/nginx:1.13.7
- Set an environment variable of NGINX PORT=8080 and also expose that port for the container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

**Answer:**

See the solution below.

Explanation:

Solution:



```
student@node-1:~$ kubectl create deployment api --image=lfccncf/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
THE **LINUX** FOUNDATION

```

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: lfccncf/nginx:1.13.7-alpine
          name: nginx
          resources: {}
status: {}
~
"nginx_deployment.yml" 25L, 421C
4,1 All

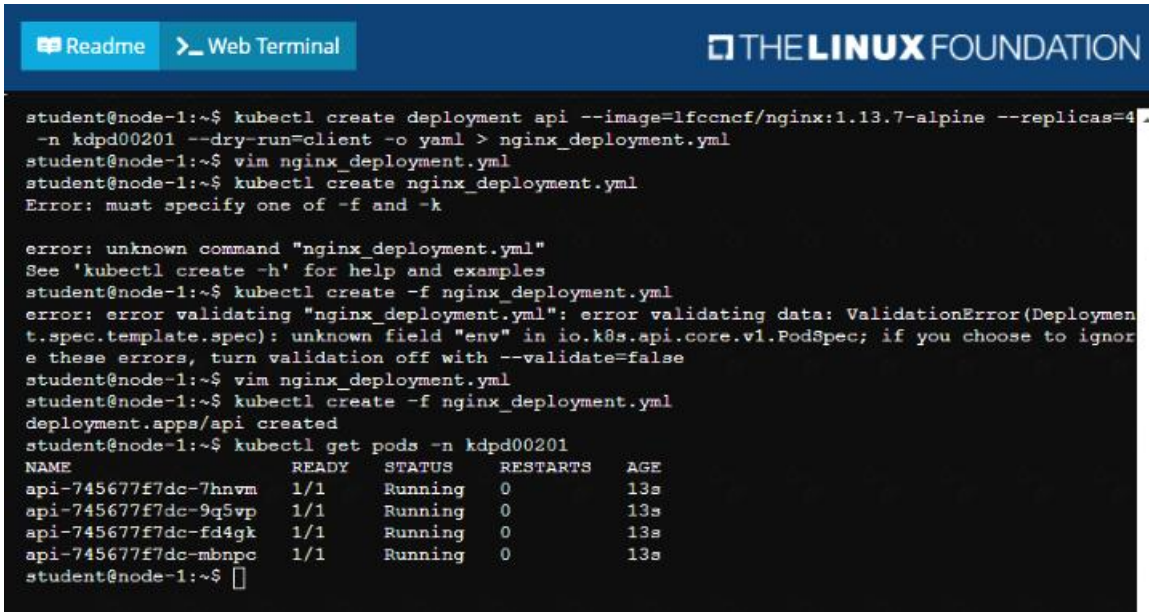
```

Readme
Web Terminal
THE **LINUX** FOUNDATION

```

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: api
  name: api
  namespace: kdpd00201
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"
~
23,8 All

```



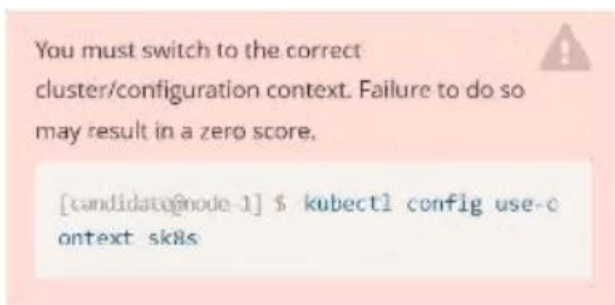
```

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
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 kdpd00201
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:~$

```

## 5. Exhibit:



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

```

## Task:

The pod for the Deployment named nosql in the craytish namespace fails to start because its container runs out of resources.

Update the nosql Deployment so that the Pod:

The nosql Deployment's manifest file can be found at  
 ~/chief-cardinal/nosql.yaml

## Answer:

See the solution below.

Explanation:

Solution:

```

candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ vim ~/chief-cardinal/nosql.yaml

```

12,1	ALL
------	-----

:WQ

```

File Edit View Terminal Tabs Help
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:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ vim ~/chief-cardinal/nosql.yaml
candidate@node-1:~$ vim ~/chief-cardinal/nosql.yaml
candidate@node-1:~$ kubectl apply -f ~/chief-cardinal/nosql.yaml
deployment.apps/nosql configured
candidate@node-1:~$ kubectl get pods -n crayfish
NAME                                READY   STATUS    RESTARTS   AGE
nosql-74ccc7d64-lkqlg               1/1     Running   0           3m2s
candidate@node-1:~$ kubectl get deploy -n crayfish
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nosql   1/1     1             1           7h16m
candidate@node-1:~$

```

## 6. Exhibit:



## 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 me pods defined by the specification of kdsn00101-deployment
- Is of type NodePort
- Has a name of cherry

**Answer:**


See the solution below.

Explanation:

Solution:

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

Readme
Web Terminal

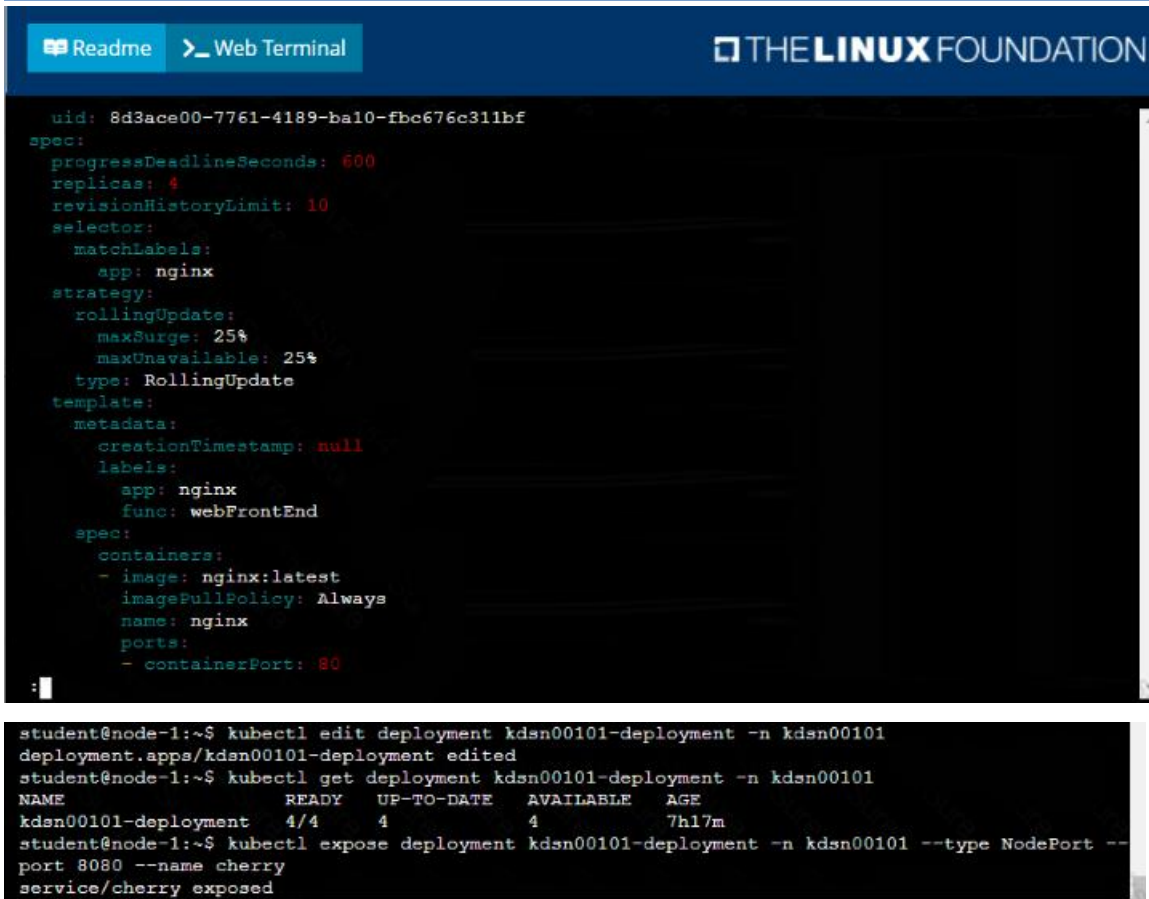


```

# 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:
    "/tmp/kubectl-edit-d4y5r.yaml" 70L, 1957C
1,1
Top

```



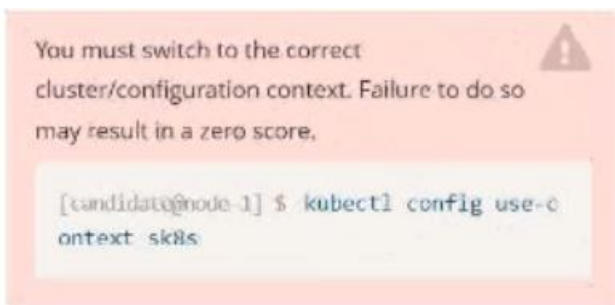


The screenshot shows a terminal window with a dark background. At the top, there's a header bar with 'Readme' and 'Web Terminal' buttons, and 'THE LINUX FOUNDATION' logo. The terminal output shows the configuration of a Kubernetes deployment named 'kdsn00101-deployment'. The configuration includes a selector for 'app: nginx', a rolling update strategy with a max surge of 25%, and a container named 'nginx' using the 'nginx:latest' image. Below the configuration, several terminal commands are executed: 'kubectl edit deployment kdsn00101-deployment -n kdsn00101', 'kubectl get deployment kdsn00101-deployment -n kdsn00101', and 'kubectl expose deployment kdsn00101-deployment -n kdsn00101 --type NodePort --port 8080 --name cherry'. The output of the 'get' command shows the deployment is in a 'READY' state with 4 replicas.

```
uid: 8d3ace00-7761-4189-ba10-fbc676c311bf
spec:
  progressDeadlineSeconds: 600
  replicas: 4
  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
```

## 7. Exhibit:



## Task:

Modify the existing Deployment named broker-deployment running in namespace quetzal so that its containers.

The broker-deployment is manifest file can be found at:

```
~/k8s-ing-mock-casiv/broker-deployment.yaml
```

## Answer:

See the solution below.

Explanation: Solution:

```
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ vim
```

Text Description automatically generated

```
File Edit View Terminal Tabs Help
containers:
- name: broker
  image: redis:alpine
  ports:
  - containerPort: 6379
  securityContext:
    runAsUser: 30000
    privileged: false

:~
:wq

candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ vim ~/daring-moccasin/broker-deployment.yaml
candidate@node-1:~$ kubectl apply -f ~/daring-moccasin/broker-deployment.yaml
deployment.apps/broker-deployment configured
candidate@node-1:~$ kubectl get pods -n quetzal
NAME                                READY   STATUS    RESTARTS   AGE
broker-deployment-65446d6d94-868p6  1/1     Running   0           30s
broker-deployment-65446d6d94-8dn7l  1/1     Running   0           32s
broker-deployment-65446d6d94-p4h4l  1/1     Running   0           31s
candidate@node-1:~$ kubectl get deploy -n quetzal
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
broker-deployment  3/3     3             3           7h3m
candidate@node-1:~$
```

8. Exhibit:



Context

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

Perform the following tasks:

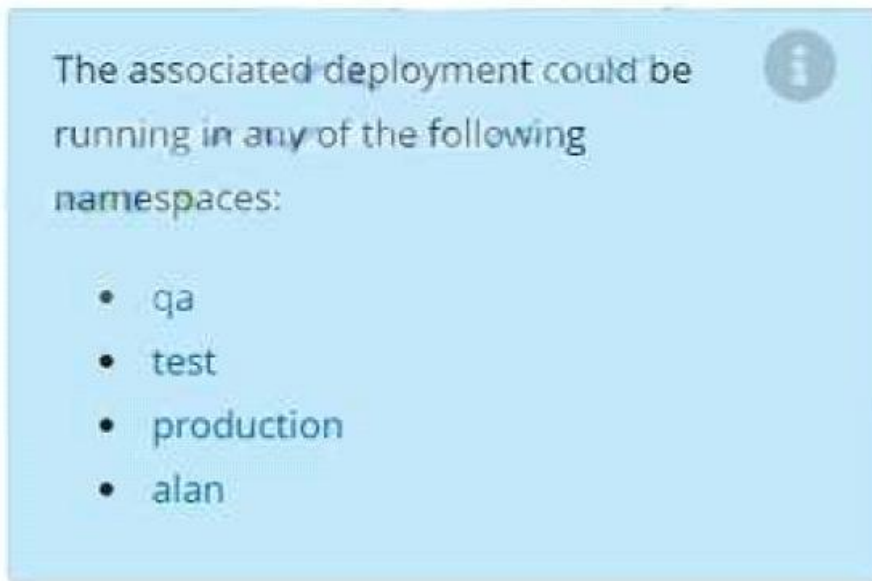
- Find the broken pod and store its name and namespace to /opt/KDOB00401/broken.txt in the format:

```
<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.



#### Answer:

See the solution below.

Explanation:

Solution:

Create the Pod: `kubectl create`

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

`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



"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 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

"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 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`

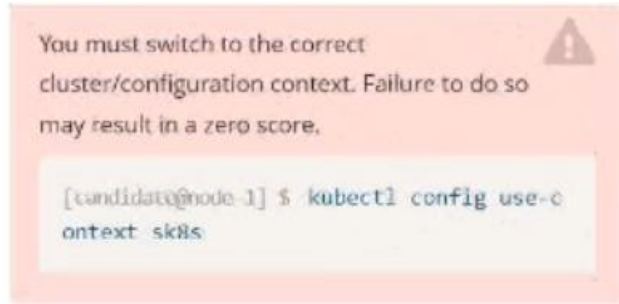
The output shows that RESTARTS has been incremented:

NAME READY STATUS RESTARTS AGE

liveness-exec 1/1 Running 1 m

9. Exhibit:





Task:

Create a Pod named nginx resources in the existing pod resources namespace. Specify a single container using nginx:stable image.

Specify a resource request of 300m cpus and 1Gi of memory for the Pod's container.

**Answer:**

See the solution below.

Explanation:

Solution:

```
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl run nginx-resources -n pod-resources --image=nginx:stable --dry-run=client -o yaml > hw.yaml
candidate@node-1:~$ vim hw.yaml
```

Text Description automatically generated with medium confidence

```
File Edit View Terminal Tabs Help
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx-resources
  name: nginx-resources
  namespace: pod-resources
spec:
  containers:
  - image: nginx:stable
    name: nginx-resources
    resources:
      requests:
        cpu: 300m
        memory: "1Gi"
```

Text Description automatically generated

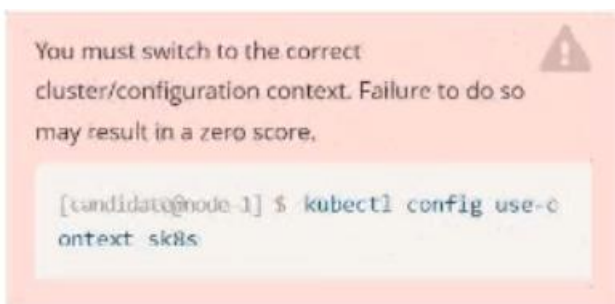


```
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl run nginx-resources -n pod-resources --image=nginx:stable --dry-run=client -o yaml > hw.yaml
candidate@node-1:~$ vim hw.yaml
candidate@node-1:~$ kubectl create -f hw.yaml
pod/nginx-resources created
candidate@node-1:~$ kubectl get pods -n pod-resources
NAME                READY   STATUS    RESTARTS   AGE
nginx-resources     1/1     Running   0           13s
candidate@node-1:~$ kubectl describe pods -n pod-resources
```

Text Description automatically generated

```
File Edit View Terminal Tabs Help
memory: 1Gi
Environment: <none>
Mounts:
  /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-dmx9j (ro)
Conditions:
  Type            Status
  Initialized      True
  Ready           True
  ContainersReady True
  PodScheduled    True
Volumes:
  kube-api-access-dmx9j:
    Type: Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName: kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI: true
QoS Class: Burstable
Node-Selectors: <none>
Tolerations:
  node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason      Age   From          Message
  ----     -
  Normal   Scheduled   20s   default-scheduler   Successfully assigned pod-resources/nginx-resources to k8s-node-0
  Normal   Pulling     19s   kubelet         Pulling image "nginx:stable"
  Normal   Pulled      13s   kubelet         Successfully pulled image "nginx:stable" in 6.55664052s
  Normal   Created     13s   kubelet         Created container nginx-resources
  Normal   Started     12s   kubelet         Started container nginx-resources
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl create deploy expose -n ckad00014 --image lfccnrf/nginx:1.13.7 --dry-run=client -o yaml>
```

10. Exhibit:



Task:

A pod within the Deployment named buffalo-deployment and in namespace gorilla is logging errors.

1) Look at the logs identify errors messages.

Find errors, including User "system:serviceaccount:gorilla:default" cannot list resource "deployment" [...] in the namespace "gorilla"

The buffalo-deployment 'S manifest can be found at -/prompt/escargot/buffalo-deployment.yaml



**Answer:**

See the solution below.

Explanation:

Solution:

Text Description automatically generated

```
File Edit View Terminal Tabs Help
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:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ vim ~/prompt-escargot/buffalo-deployment.yaml
candidate@node-1:~$ vim ~/prompt-escargot/buffalo-deployment.yaml
candidate@node-1:~$ kubectl apply -f ~/prompt-escargot/buffalo-deployment.yaml
deployment.apps/buffalo-deployment configured
candidate@node-1:~$ kubectl get pods -n gorilla
NAME                                READY   STATUS              RESTARTS   AGE
buffalo-deployment-776844df7f-r5fsb 1/1     Running             0           6h38m
buffalo-deployment-859898c6f5-zx5gj 0/1     ContainerCreating   0           8s
candidate@node-1:~$ kubectl get deploy -n gorilla
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
buffalo-deployment 1/1      1             1           6h38m
candidate@node-1:~$
```

11. Exhibit:



Context

Developers occasionally need to submit pods that run periodically. Task

Follow the steps below to create a pod that will start at a predetermined time and]which runs to completion only once each time it is started:

- Create a YAML formatted Kubernetes manifest /opt/KDPD00301/periodic.yaml that runs the following shell command: date in a single busybox container. The command should run every minute and must complete within 22 seconds or be terminated by Kubernetes. The Cronjob name and container name

should both be hello

- Create the resource in the above manifest and verify that the job executes successfully at least once

**Answer:**

See the solution below.

Explanation:

Solution:

```

Readme Web Terminal THE LINUX FOUNDATION

student@node-1:~$ kubectl create cronjob hello --image=busybox --schedule "*" * * * * --dry-run=
client -o yaml > /opt/KDPD00301/periodic.yaml
error: unable to match a printer suitable for the output format "yaml", allowed formats are: go-t
emplate, go-template-file, json, jsonpath, jsonpath-as-json, jsonpath-file, name, template, templatefile
, yaml
student@node-1:~$ kubectl create cronjob hello --image=busybox --schedule "*" * * * * --dry-run=
client -o yaml > /opt/KDPD00301/periodic.yaml
student@node-1:~$ vim /opt/KDPD00301/periodic.yaml

```

```

Readme Web Terminal THE LINUX FOUNDATION

apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  jobTemplate:
    metadata:
      name: hello
    spec:
      template:
        spec:
          containers:
            - image: busybox
              name: hello
              args: ["/bin/sh", "-c", "date"]
              restartPolicy: Never
          schedule: */1 * * * *
          startingDeadlineSeconds: 22
          concurrencyPolicy: Allow

```

```

Readme Web Terminal THE LINUX FOUNDATION

student@node-1:~$ kubectl create cronjob hello --image=busybox --schedule "*" * * * * --dry-run=
client -o yaml > /opt/KDPD00301/periodic.yaml
error: unable to match a printer suitable for the output format "yaml", allowed formats are: go-t
emplate, go-template-file, json, jsonpath, jsonpath-as-json, jsonpath-file, name, template, templatefile
, yaml
student@node-1:~$ kubectl create cronjob hello --image=busybox --schedule "*" * * * * --dry-run=
client -o yaml > /opt/KDPD00301/periodic.yaml
student@node-1:~$ vim /opt/KDPD00301/periodic.yaml
student@node-1:~$ kubectl create -f /opt/KDPD00301/periodic.yaml
cronjob.batch/hello created
student@node-1:~$ kubectl get cronjob
NAME      SCHEDULE      SUSPEND   ACTIVE   LAST SCHEDULE   AGE
hello     */1 * * * *   False    0        <none>           6s
student@node-1:~$

```



## 12. Exhibit:

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

### Context

A container within the poller pod is hard-coded to connect the nginxsvc service on port 90. As this port changes to 5050 an additional container needs to be added to the poller pod which adapts the container to connect to this new port. This should be realized as an ambassador container within the pod.

### Task



- Update the nginxsvc service to serve on port 5050.
- Add an HAProxy container named haproxy bound to port 90 to the poller pod and deploy the enhanced pod. Use the image haproxy and inject the configuration located at /opt/KDMC00101/haproxy.cfg, with a ConfigMap named haproxy-config, mounted into the container so that haproxy.cfg is available at /usr/local/etc/haproxy/haproxy.cfg. Ensure that you update the args of the poller container to connect to localhost instead of nginxsvc so that the connection is correctly proxied to the new service endpoint. You must not modify the port of the endpoint in poller's args . The spec file used to create the initial poller pod is available in /opt/KDMC00101/poller.yaml

### Answer:

See the solution below.

### Explanation:

### Solution:

apiVersion: apps/v1 kind: Deployment metadata:

name: my-nginx spec:

selector:

matchLabels: run: my-nginx replicas: 2 template: metadata: labels:

run: my-nginx spec: containers:

- name: my-nginx image: nginx ports:

- containerPort: 90



This makes it accessible from any node in your cluster. Check the nodes the Pod is running on: `kubectl`

```
apply -f ./run-my-nginx.yaml
```

```
kubectl get pods -l run=my-nginx -o wide
```

```
NAME READY STATUS RESTARTS AGE IP NODE
```

```
my-nginx-3800858182-jr4a2 1/1 Running 0 13s 10.244.3.4 kubernetes-minion-905m
```

```
my-nginx-3800858182-kna2y 1/1 Running 0 13s 10.244.2.5 kubernetes-minion-ljyd
```

Check your pods' IPs:

```
kubectl get pods -l run=my-nginx -o yaml | grep podIP podIP: 10.244.3.4
```

```
podIP: 10.244.2.5
```

13. Exhibit:



Context

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

Task

Update the app-a deployment in the production namespace to run as the restrictedservice service account.

The service account has already been created.

**Answer:**

See the solution below.



Explanation:

Solution:

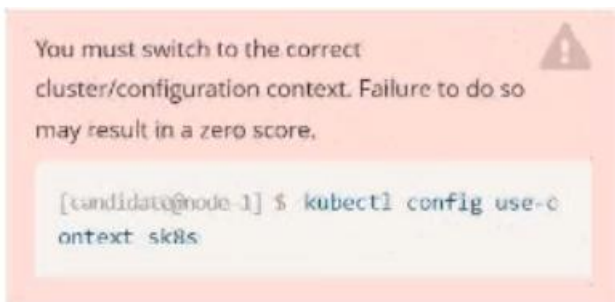


```

student@node-1:~$ kubectl get serviceaccount -n production
NAME          SECRETS  AGE
default       1        6h46m
restrictedservice 1        6h46m
student@node-1:~$ kubectl get deployment -n production
NAME    READY  UP-TO-DATE  AVAILABLE  AGE
app-a   3/3    3           3          6h46m
student@node-1:~$ kubectl set serviceaccount deployment app-a restrictedservice -n production
deployment.apps/app-a serviceaccount updated
student@node-1:~$ 

```

14. Exhibit:



Task

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

The Deployment's manifest files can be found at

`~/spicy-pikachu/backend-deployment.yaml`.

Modify the Deployment specifying a readiness probe using path `/healthz`.

Set `initialDelaySeconds` to 8 and `periodSeconds` to 5.



**Answer:**

See the solution below.

Explanation: Solution:

```
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:~$ vim ~/spicy-pikachu/backend-deployment.yaml
```

Text Description automatically generated

```
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 --
```

26/28

Top

```
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:~$ 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
```

15. Exhibit:



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

You can access `sk8s-node-0` by issuing the following command:

```
[student@node-1] % | ssh sk8s-node-0
```

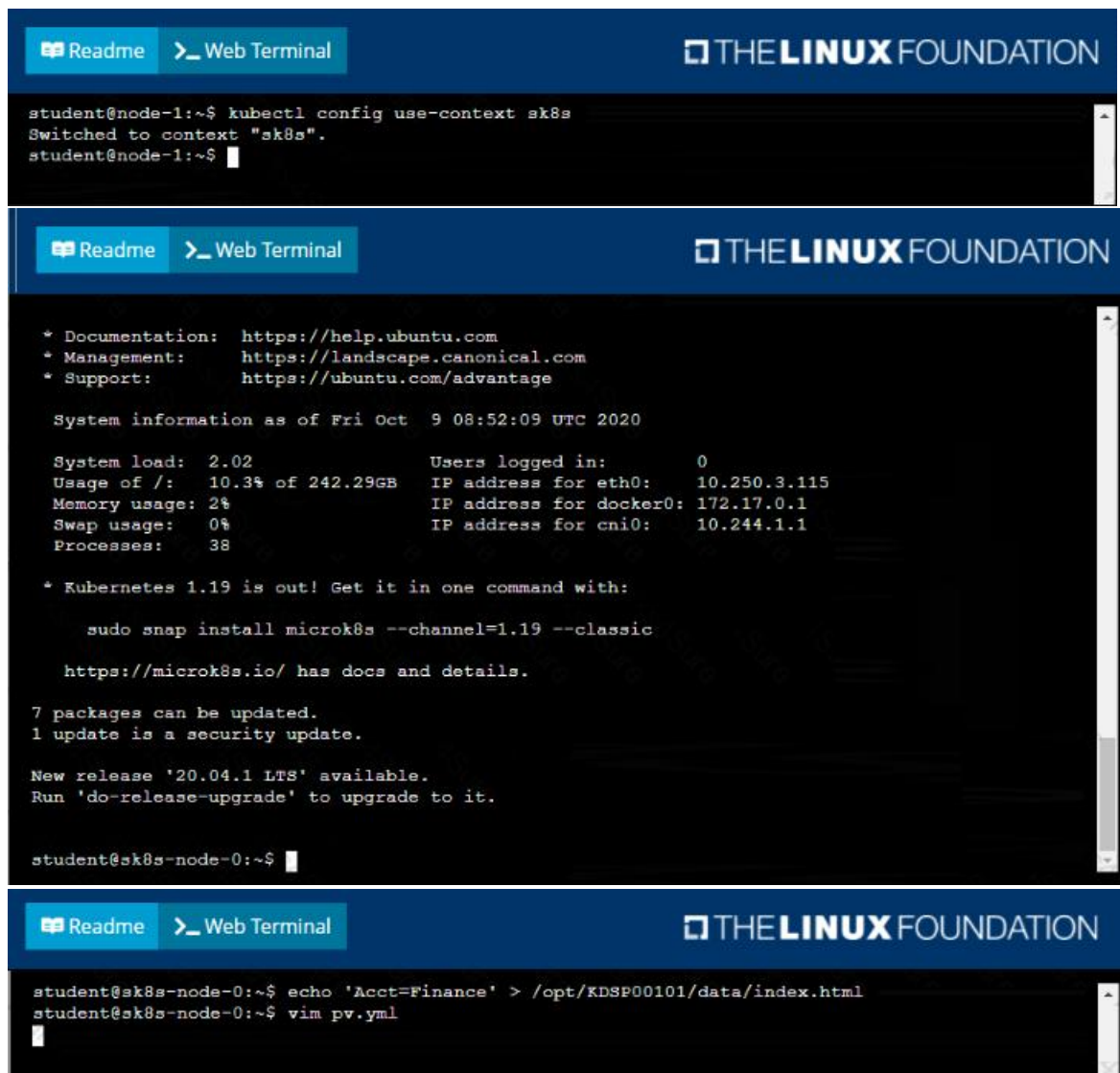
Ensure that you return to the base node (with hostname `node-1`) once you have completed your work on `sk8s-node-0` 

### Answer:

See the solution below.

Explanation:

Solution:



The screenshots show a web terminal interface with a blue header containing 'THE LINUX FOUNDATION' logo and navigation tabs for 'Readme' and 'Web Terminal'.

**Screenshot 1:** Shows the terminal output of the command `kubect1 config use-context sk8s`. The output is:
 

```
student@node-1:~$ kubect1 config use-context sk8s
Switched to context "sk8s".
student@node-1:~$
```

**Screenshot 2:** Shows system information and update notifications. The output includes:
 

```
* 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:~$
```

**Screenshot 3:** Shows the terminal output of the command `echo 'Acct=Finance' > /opt/KDSP00101/data/index.html` and the command `vim pv.yml`. The output is:
 

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

Readme
Web Terminal
THE **LINUX** FOUNDATION

```

-- INSERT --
0,1 All

```

Readme
Web Terminal
THE **LINUX** FOUNDATION

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: storage
  hostPath:
    path: /opt/KDSP00101/data
    type: Directory

```

Readme
Web Terminal
THE **LINUX** FOUNDATION

```

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

```



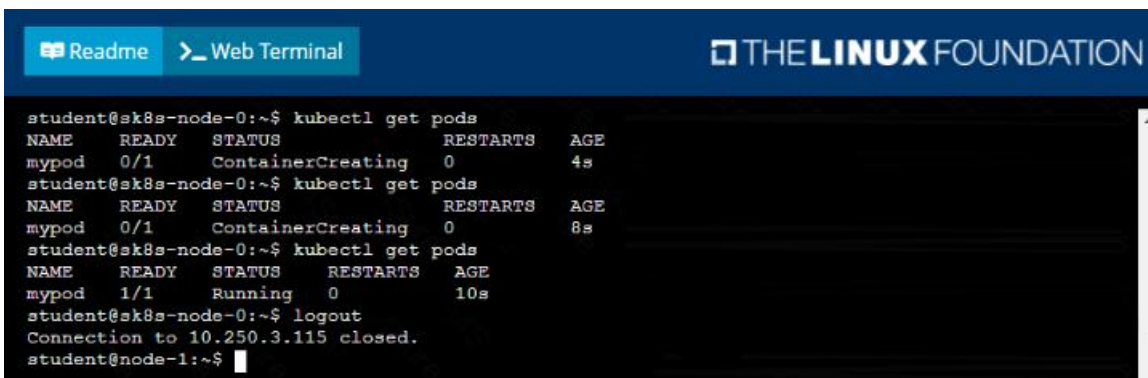


```

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-clai
  
```

```

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:~$
  
```

16. Exhibit:



Context

You are tasked to create a ConfigMap and consume the ConfigMap in a pod using a volume mount. Task  
Please complete the following:

- Create a ConfigMap named another-config containing the key/value pair: key4/value3
- start a pod named nginx-configmap containing a single container using the nginx image, and mount the key you just created into the pod under directory /also/a/path

**Answer:**

See the solution below.

Explanation:

Solution:

```
student@node-1:~$ kubectl create configmap another-config --from-literal=key4=value3
configmap/another-config created
student@node-1:~$ kubectl get configmap
NAME          DATA   AGE
another-config 1       5s
student@node-1:~$ kubectl run nginx-configmap --image=nginx --dry-run=client -o yaml > nginx_configmap.yaml
student@node-1:~$ vim nginx_configmap.yaml ^C
student@node-1:~$ mv nginx_configmap.yaml nginx_configmap.yaml
student@node-1:~$ vim nginx_co
```

Readme Web Terminal THE LINUX FOUNDATION

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx-configmap
  name: nginx-configmap
spec:
  containers:
  - image: nginx
    name: nginx-configmap
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}

"nginx_configmap.yaml" 15L, 262C 1,1 All
```

Readme
Web Terminal
THE LINUX FOUNDATION

```

apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx-configmap
  name: nginx-configmap
spec:
  containers:
  - image: nginx
    name: nginx-configmap
    volumeMounts:
    - name: myvol
      mountPath: /also/a/path
  volumes:
  - name: myvol
    configMap:
      name: another-config
~
~
~
~
~
~
~
~
~
~
13,6
All

```

```

student@node-1:~$ kubectl create configmap another-config --from-literal=key4=value3
configmap/another-config created
student@node-1:~$ kubectl get configmap
NAME          DATA   AGE
another-config 1       5s
student@node-1:~$ kubectl run nginx-configmap --image=nginx --dry-run=client -o yaml > nginx_conf
igmap.yaml
student@node-1:~$ vim nginx_configmap.yaml ^C
student@node-1:~$ mv nginx_configmap.yaml nginx_configmap.yaml
student@node-1:~$ vim nginx_configmap.yaml
student@node-1:~$

```

```

student@node-1:~$ kubectl run nginx-configmap --image=nginx --dry-run=client -o yaml > nginx_conf
igmap.yaml
student@node-1:~$ vim nginx_configmap.yaml ^C
student@node-1:~$ mv nginx_configmap.yaml nginx_configmap.yaml
student@node-1:~$ vim nginx_configmap.yaml
student@node-1:~$ kubectl create f nginx_configmap.yaml
Error: must specify one of -f and -k

error: unknown command "f nginx_configmap.yaml"
See 'kubectl create -h' for help and examples
student@node-1:~$ kubectl create -f nginx_configmap.yaml
error: error validating "nginx_configmap.yaml": error validating data: ValidationError(Pod.spec.c
ontainers[1]): unknown field "mountPath" in io.k8s.api.core.v1.Container; if you choose to ignor
e these errors, turn validation off with --validate=false
student@node-1:~$ vim nginx_configmap.yaml

```

Readme
Web Terminal

```

student@node-1:~$ kubectl create f nginx_configmap.yml
Error: must specify one of -f and -k

error: unknown command "f nginx_configmap.yml"
See 'kubectl create -h' for help and examples
student@node-1:~$ kubectl create -f nginx_configmap.yml
error: error validating "nginx_configmap.yml": error validating data: ValidationError(Pod.spec.con
ainers[1]): unknown field "mountPath" in io.k8s.api.core.v1.Container; if you choose to ignor
e these errors, turn validation off with --validate=false
student@node-1:~$ vim nginx_configmap.yml
student@node-1:~$ kubectl create -f nginx_configmap.yml
pod/nginx-configmap created
student@node-1:~$ kubectl get pods
NAME                READY   STATUS              RESTARTS   AGE
liveness-http       1/1     Running             0           6h44m
nginx-101           1/1     Running             0           6h45m
nginx-configmap     0/1     ContainerCreating   0           5s
nginx-secret        1/1     Running             0           5m39s
poller              1/1     Running             0           6h44m
student@node-1:~$ kubectl get pods
NAME                READY   STATUS   RESTARTS   AGE
liveness-http       1/1     Running  0           6h44m
nginx-101           1/1     Running  0           6h45m
nginx-configmap     1/1     Running  0           8s
nginx-secret        1/1     Running  0           5m42s
poller              1/1     Running  0           6h45m
student@node-1:~$

```

17. Exhibit:

No configuration context change is required for this task.

Task:

A Dockerfile has been prepared at `-/human-stork/build/Dockerfile`

Multiple image builders and tools have been pre-installed in the base system, including: docker, skopeo, buildah, img, and podman.

Please do not push the built image to a registry, run a container, or otherwise consume it.

**Answer:**

See the solution below.

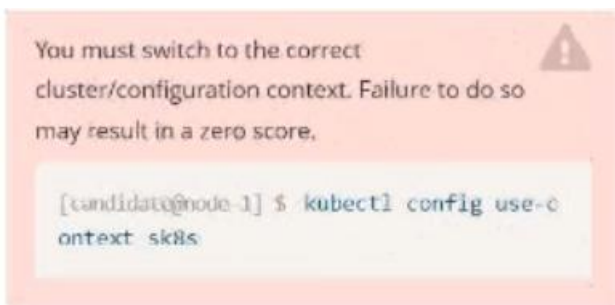
Explanation: Solution:

```

candidate@node-1:~$ cd humane-stork/build/
candidate@node-1:~/humane-stork/build$ ls -l
total 16
-rw-r--r-- 1 candidate candidate 201 Sep 24 04:21 Dockerfile
-rw-r--r-- 1 candidate candidate 644 Sep 24 04:21 text1.html
-rw-r--r-- 1 candidate candidate 813 Sep 24 04:21 text2.html
-rw-r--r-- 1 candidate candidate 383 Sep 24 04:21 text3.html
candidate@node-1:~/humane-stork/build$ sudo docker build -t macaque:3.0 .
Sending build context to Docker daemon 6.144kB
Step 1/5 : FROM docker.io/lfcncf/nginx:mainline
--> ea335eeal7ab
Step 2/5 : ADD text1.html /usr/share/nginx/html/
--> 8967ee9ee5d0
Step 3/5 : ADD text2.html /usr/share/nginx/html/
--> cb0554422f26
Step 4/5 : ADD text3.html /usr/share/nginx/html/
--> 62e879ab821e
Step 5/5 : COPY text2.html /usr/share/nginx/html/index.html
--> 331c8a94372c
Successfully built 331c8a94372c
Successfully tagged macaque:3.0
candidate@node-1:~/humane-stork/build$ sudo docker save macaque:3.0 > ~/humane-stork/macaque-3.0.tar
candidate@node-1:~/humane-stork/build$ cd ..
candidate@node-1:~/humane-stork$ ls -l
total 142532
drwxr-xr-x 2 candidate candidate 4096 Sep 24 04:21 build
-rw-rw-r-- 1 candidate candidate 145948672 Sep 24 11:39 macaque-3.0.tar
candidate@node-1:~/humane-stork$

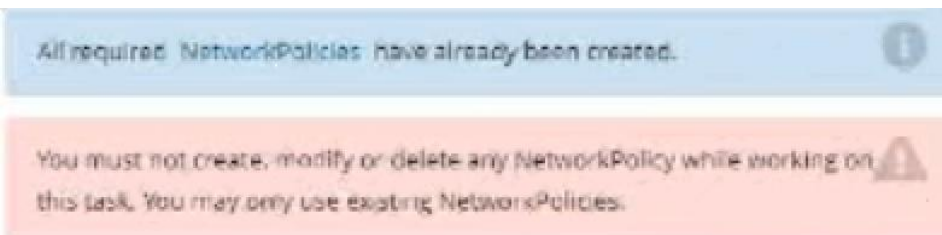
```

18. Exhibit:



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:

See the solution below.

Explanation:

Solution:

```

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

```



```

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:~$

```

19. Exhibit:



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:**

See the solution below.

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; sleep 30; rm -rf /tmp/healthy; sleep 600

livenessProbe: exec: command:

- cat

- /tmp/healthy

initialDelaySeconds: 5

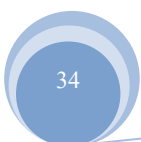
periodSeconds: 5

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 co





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

20. Exhibit:

```
Set configuration context:

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



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

**Answer:**

See the solution below.

Explanation:

Solution:

```
student@node-1:~$ kubectl run nginx-resources -n pod-resources --image=nginx --dry-run=client -o
yaml > nginx_resources.yml
student@node-1:~$ vim nginx_
```

[illegible]

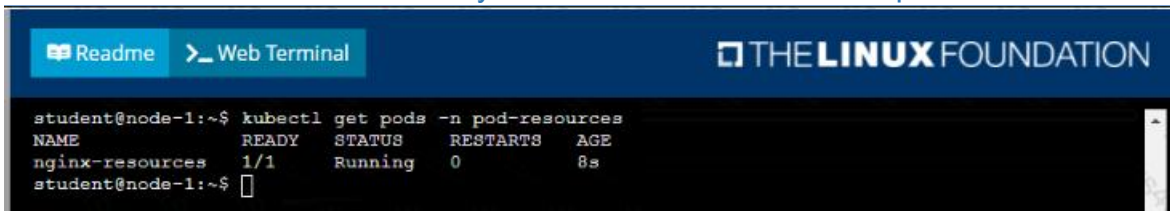
Readme

Web Terminal

THE **LINUX** FOUNDATION

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx-resources
    name: nginx-resources
    namespace: pod-resources
spec:
  containers:
    - image: nginx
      name: nginx-resources
      resources:
        requests:
          cpu: 200m
          memory: "1Gi"
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

15,22All



```

student@node-1:~$ kubectl get pods -n pod-resources
NAME          READY   STATUS    RESTARTS   AGE
nginx-resources 1/1     Running   0           8s
student@node-1:~$
    
```

21. Exhibit:



```

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



Context

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

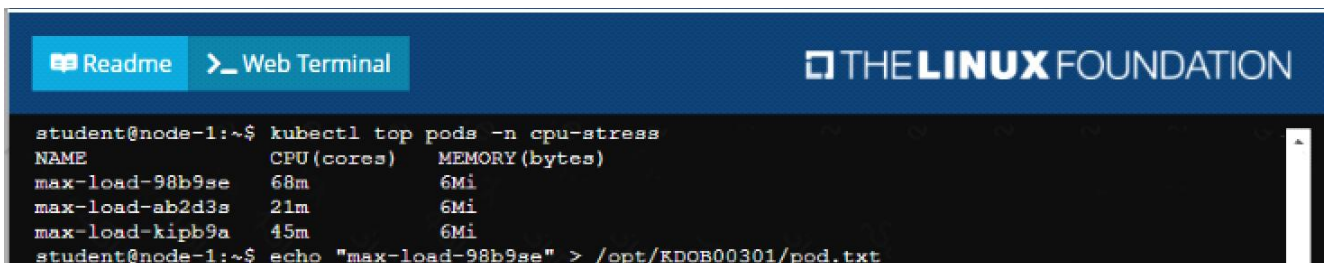
- From the pods running in namespace cpu-stress , write the name only of the pod that is consuming the most CPU to file /opt/KDOBG030I/pod.txt, which has already been created.

**Answer:**

See the solution below.

Explanation:

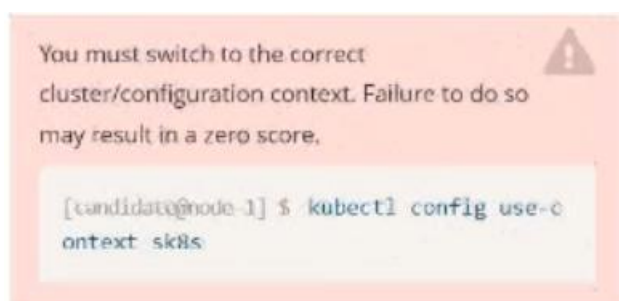
Solution:



```

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/KDOBG0030I/pod.txt
    
```

22. Exhibit:



```

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 k8s
    
```





Task:

- To run 2 replicas of the pod
- Add the following label on the pod:

Role userUI

**Answer:**

See the solution below.

Explanation:

Solution:

Text Description automatically generated

```
File Edit View Terminal Tabs Help
# reopened with the relevant failures.
#
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: "2022-09-24T04:27:03Z"
  generation: 1
  labels:
    app: nginx
  name: ckad00017-deployment
  namespace: ckad00017
  resourceVersion: "3349"
  uid: lcd67613-fade-46e9-b741-94298b9c6e7c
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
-- INSERT --
```

Text Description automatically generated



```
File Edit View Terminal Tabs Help
name: ckad00017-deployment
namespace: ckad00017
resourceVersion: "3349"
uid: 1cd67613-fade-46e9-b741-94298b9c6e7c
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
        role: userUI
    spec:
      containers:
      - image: nginx:latest
        imagePullPolicy: Always
        name: nginx
        ports:
        - containerPort: 80
          protocol: TCP
      resources: {}
-- INSERT --
```

35,21 33%

Text Description automatically generated

```
File Edit View Terminal Tabs Help
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:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ vim ~/prompt-escargot/buffalo-deployment.yaml
candidate@node-1:~$ vim ~/prompt-escargot/buffalo-deployment.yaml
candidate@node-1:~$ kubectl apply -f ~/prompt-escargot/buffalo-deployment.yaml
deployment.apps/buffalo-deployment configured
candidate@node-1:~$ kubectl get pods -n gorilla
NAME READY STATUS RESTARTS AGE
buffalo-deployment-776844df7f-r5fsb 1/1 Running 0 6h38m
buffalo-deployment-859898c6f5-zx5gj 0/1 ContainerCreating 0 8s
candidate@node-1:~$ kubectl get deploy -n gorilla
NAME READY UP-TO-DATE AVAILABLE AGE
buffalo-deployment 1/1 1 1 6h38m
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl edit deploy ckad00017-deployment -n ckad00017
deployment.apps/ckad00017-deployment edited
candidate@node-1:~$
```

```

File Edit View Terminal Tabs Help
candidate@node-1:~$ kubectl get pods -n gorilla
NAME                                READY    STATUS    RESTARTS   AGE
buffalo-deployment-776844df7f-r5fsb 1/1      Running   0           6h38m
buffalo-deployment-859898c6f5-zx5gj 0/1      ContainerCreating 0           8s
candidate@node-1:~$ kubectl get deploy -n gorilla
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
buffalo-deployment 1/1      1              1            6h38m
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl edit deploy ckad00017-deployment -n ckad00017
deployment.apps/ckad00017-deployment edited
candidate@node-1:~$ kubectl expose deploy ckad00017-deployment -n ckad00017
ckad00014 ckad00015 ckad00017
candidate@node-1:~$ kubectl expose deploy ckad00017-deployment -n ckad00017
ckad00014 ckad00015 ckad00017
candidate@node-1:~$ kubectl expose deploy ckad00017-deployment -n ckad00017
ckad00014 ckad00015 ckad00017
candidate@node-1:~$ kubectl expose deploy ckad00017-deployment -n ckad00017
ckad00014 ckad00015 ckad00017
candidate@node-1:~$ kubectl expose deploy ckad00017-deployment -n ckad00017
ckad00014 ckad00015 ckad00017
candidate@node-1:~$ kubectl expose deploy ckad00017-deployment -n ckad00017
ckad00014 ckad00015 ckad00017
candidate@node-1:~$ kubectl expose deploy ckad00017-deployment -n ckad00017
ckad00014 ckad00015 ckad00017
candidate@node-1:~$ kubectl expose deploy ckad00017-deployment -n ckad00017 --name=cherry --port=8888 --type=NodePort
service/cherry exposed
candidate@node-1:~$

candidate@node-1:~$ kubectl get svc
NAME                TYPE                CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes           ClusterIP           10.96.0.1     <none>         443/TCP    77d
candidate@node-1:~$ kubectl get svc -n ckad00017
NAME                TYPE                CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
cherry              NodePort            10.100.100.176 <none>         8888:30683/TCP 24s
candidate@node-1:~$ kubectl expose service deploy ckad00017-deployment -n ckad00017 --name=cherry --port=8888 --type=NodePort
Error from server (NotFound): services "deploy" not found
Error from server (NotFound): services "ckad00017-deployment" not found
candidate@node-1:~$ kubectl get svc -n ckad00017
NAME                TYPE                CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
cherry              NodePort            10.100.100.176 <none>         8888:30683/TCP 46s
candidate@node-1:~$

File Edit View Terminal Tabs Help
candidate@node-1:~$ kubectl expose service deploy ckad00017-deployment -n ckad00017 --name=cherry --port=8888 --type=NodePort
Error from server (NotFound): services "deploy" not found
Error from server (NotFound): services "ckad00017-deployment" not found
candidate@node-1:~$ kubectl get svc -n ckad00017
NAME                TYPE                CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
cherry              NodePort            10.100.100.176 <none>         8888:30683/TCP 46s
candidate@node-1:~$ history
 1 vi ~/spicy-pikachu/backend-deployment.yaml
 2 kubectl config use-context sk8s
 3 vim .vimrc
 4 vim ~/spicy-pikachu/backend-deployment.yaml
 5 kubectl apply -f ~/spicy-pikachu/backend-deployment.yaml
 6 kubectl get pods -n staging
 7 kubectl get deploy -n staging
 8 vim ~/spicy-pikachu/backend-deployment.yaml
 9 kubectl config use-context k8s
10 kubectl set serviceaccount deploy app-1 app -n frontend
11 kubectl config use-context k8s
12 vim ~/prompt-escargot/buffalo-deployment.yaml
13 kubectl apply -f ~/prompt-escargot/buffalo-deployment.yaml
14 kubectl get pods -n gorilla
15 kubectl get deploy -n gorilla
16 kubectl config use-context k8s
17 kubectl edit deploy ckad00017-deployment -n ckad00017
18 kubectl expose deploy ckad00017-deployment -n ckad00017 --name=cherry --port=8888 --type=NodePort
19 kubectl get svc
20 kubectl get svc -n ckad00017
21 kubectl expose service deploy ckad00017-deployment -n ckad00017 --name=cherry --port=8888 --type=NodePort
22 kubectl get svc -n ckad00017
23 history
candidate@node-1:~$

```

23. Exhibit:



```
Set configuration context:

[student@node-1] $ | kubectl config
use-context nk8s
```



Task

A deployment is falling on the cluster due to an incorrect image being specified. Locate the deployment, and fix the problem.

**Answer:**

See the solution below

Explanation:

create deploy hello-deploy --image=nginx --dry-run=client -o yaml > hello-deploy.yaml

Update deployment image to nginx:1.17.4: kubec

nginx=nginx:1.17.4

24. Exhibit:



```
Set configuration context:

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



Context

You sometimes need to observe a pod's logs, and write those logs to a file for further analysis. Task

Please complete the following;

- Deploy the counter pod to the cluster using the provided YAMLSpec file at /opt/KDOB00201/counter.yaml
- Retrieve all currently available application logs from the running pod and store them in the file /opt/KDOB00201/log\_Output.txt, which has already been created

**Answer:**

See the solution below.





Explanation:

Solution:

```
student@node-1:~$ kubectl create -f /opt/KDOB00201/counter.yaml
pod/counter created
student@node-1:~$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
counter       1/1     Running   0           10s
liveness-http 1/1     Running   0           6h45m
nginx-101     1/1     Running   0           6h46m
nginx-configmap 1/1     Running   0           107s
nginx-secret  1/1     Running   0           7m21s
poller        1/1     Running   0           6h46m
student@node-1:~$ kubectl logs counter
1: 2b305101817ae25ca60ae46510fb6d11
2: 3648cf2eae95ab680dba8f195f891af4
3: 65c8bbd4dbf70bf81f2a0984a3a44ede
4: 40d3a9c8e46f5533bb4828fbc5c8d038
5: 390442d2530a90c3602901e3fe999ac8
6: b71d95187417e139effb33af77681040
7: 66a8e55a6491e756d2d0549ad6ab90a7
8: ff2b3d583b64125d2f9129c443bb37ff
9: b6c6a12b6e77944ed8baaaf6c242dae4
10: bfec9a894a0604fc4b814b37d0a200a4
student@node-1:~$ kubectl logs counter > /opt/KDOB00201/log_output.txt
student@node-1:~$
```

```
student@node-1:~$ kubectl logs counter > /opt/KDOB00201/log_output.txt
student@node-1:~$ kubectl logs counter > /opt/KDOB00201/log_output.txt
student@node-1:~$ cat /opt/KDOB00201/log_output.txt
```

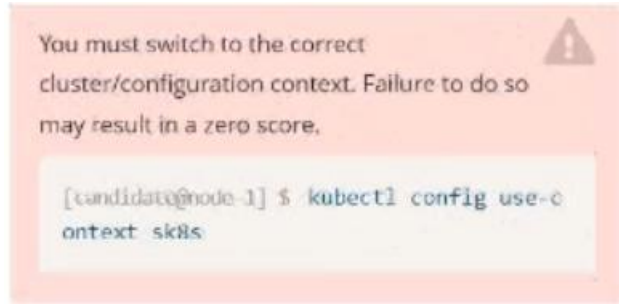
Readme

Web Terminal

THE LINUX FOUNDATION

```
student@node-1:~$ kubectl logs counter > /opt/KDOB00201/log_output.txt
student@node-1:~$ cat /opt/KDOB00201/log_output.txt
1: 2b305101817ae25ca60ae46510fb6d11
2: 3648cf2eae95ab680dba8f195f891af4
3: 65c8bbd4dbf70bf81f2a0984a3a44ede
4: 40d3a9c8e46f5533bb4828fbc5c8d038
5: 390442d2530a90c3602901e3fe999ac8
6: b71d95187417e139effb33af77681040
7: 66a8e55a6491e756d2d0549ad6ab90a7
8: ff2b3d583b64125d2f9129c443bb37ff
9: b6c6a12b6e77944ed8baaaf6c242dae4
10: bfec9a894a0604fc4b814b37d0a200a4
11: 5493cd16a1790a5fb9512b0c9d4c5dd1
12: 03f169e93e6143438e6dfe4ecb3cc9ed
13: 764b37fe611373c42d0b47154041f6eb
14: 1a56fbc1896b0ee6394136166281839e
15: ecc492eb17715de090c47345a98d98d3
16: 7974a6bec0fb44b6b8bbfc71aa3fbc74
17: 9ae01bef01748b12cc9f97a5f9f72cd6
18: 23fb22ee34d4272e4c9e005f1774515f
19: ec7e1a5d314da9a0ad45d53be5a7acae
20: 0bccdd8ee02cd42029e8162cd1c1197c
21: d6851ea43546216b95bcb81ced997102
22: 7ed9a38ea8bf0d86206569481442af44
23: 29b8416ddc63dbfcb987ab3c8198e9fe
24: 1f2062001df51a108ab25010f506716f
student@node-1:~$
```

25. Exhibit:



Task:

- 1- Update the Propertunel scaling configuration of the Deployment web1 in the ckad00015 namespace setting maxSurge to 2 and maxUnavailable to 59
- 2- Update the web1 Deployment to use version tag 1.13.7 for the lfconf/nginx container image. 3- Perform a rollback of the web1 Deployment to its previous version

**Answer:**

See the solution below.

Explanation: Solution:

```
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl edit deploy web1 -n ckad00015
```

Text Description automatically generated

```
File Edit View Terminal Tabs Help
app: nginx
strategy:
  rollingUpdate:
    maxSurge: 2%
    maxUnavailable: 5%
    type: RollingUpdate
template:
  metadata:
    creationTimestamp: null
    labels:
      app: nginx
  spec:
    containers:
      - image: lfccncf/nginx:1.13.7
        imagePullPolicy: IfNotPresent
        name: nginx
        ports:
          - containerPort: 80
            protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext: {}
    terminationGracePeriodSeconds: 30
status:
  availableReplicas: 2
  conditions:
    - lastTransitionTime: "2022-09-24T04:26:41Z"
```

```

Switched to context "k8s".
candidate@node-1:~$ kubectl create secret generic app-secret -n default --from-literal=key3=value1
secret/app-secret created
candidate@node-1:~$ kubectl get secrets
NAME      TYPE      DATA      AGE
app-secret Opaque    1          4s
candidate@node-1:~$ kubectl run nginx-secret -n default --image=nginx:stable --dry-run=client -o yaml > sec.yaml
candidate@node-1:~$ vim sec.yaml
candidate@node-1:~$ kubectl create -f sec.yaml
pod/nginx-secret created
candidate@node-1:~$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
nginx-secret        1/1     Running   0           7s
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl edit deploy web1 -n ckad00015
deployment.apps/web1 edited
candidate@node-1:~$ kubectl rollout status deploy web1 -n ckad00015
deployment "web1" successfully rolled out
candidate@node-1:~$ kubectl rollout undo deploy web1 -n ckad00015
deployment.apps/web1 rolled back
candidate@node-1:~$ kubectl rollout history deploy web1 -n ckad00015
deployment.apps/web1
REVISION  CHANGE-CAUSE
2          <none>
3          <none>
candidate@node-1:~$ kubectl get rs -n ckad00015
NAME                DESIRED   CURRENT   READY   AGE
web1-56f98bcb79     0         0         0       63s
web1-85775b6b79     2         2         2       6h53m
candidate@node-1:~$

```

26. Exhibit:



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:**

See the solution below.



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

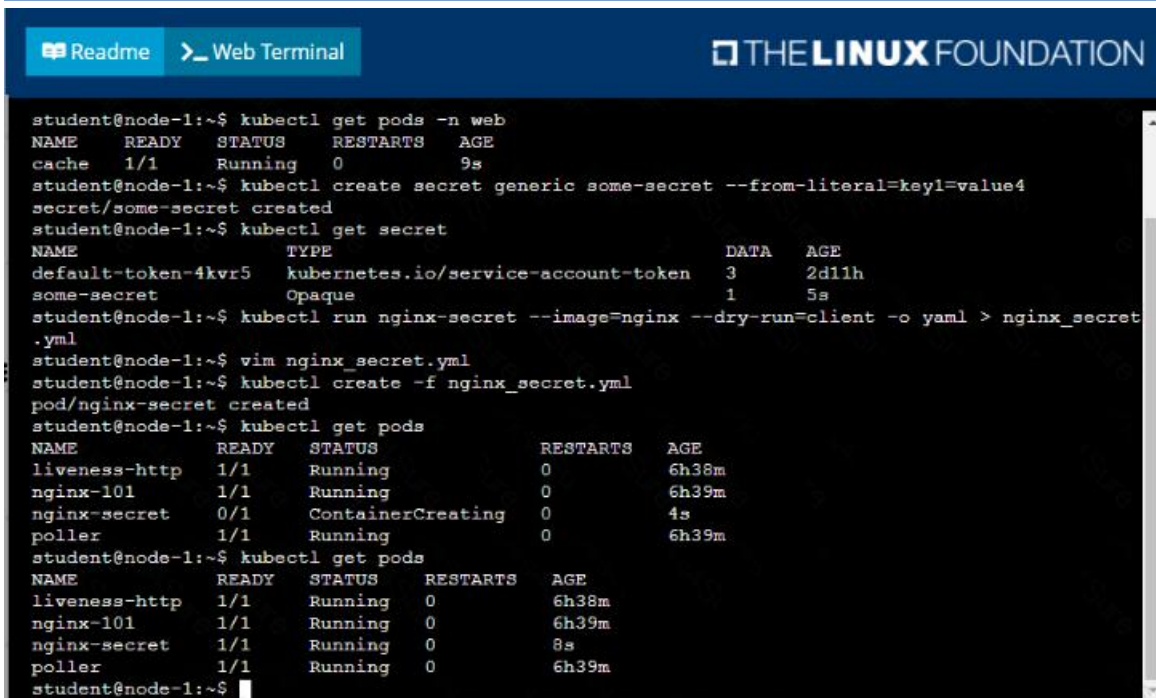
```

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
~
~
~
~
~
~
~
~
~
~
-- INSERT --
16,20
All

```



```

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=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.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:~$

```

## 27. 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/pod1.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

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

Answer:

See the solution below.

Explanation:

**Solution:**

```
student@node-1:~$ kubectl run appl --image=lfcncf/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: {}
```

Readme

Web Terminal

THE **LINUX** FOUNDATION

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

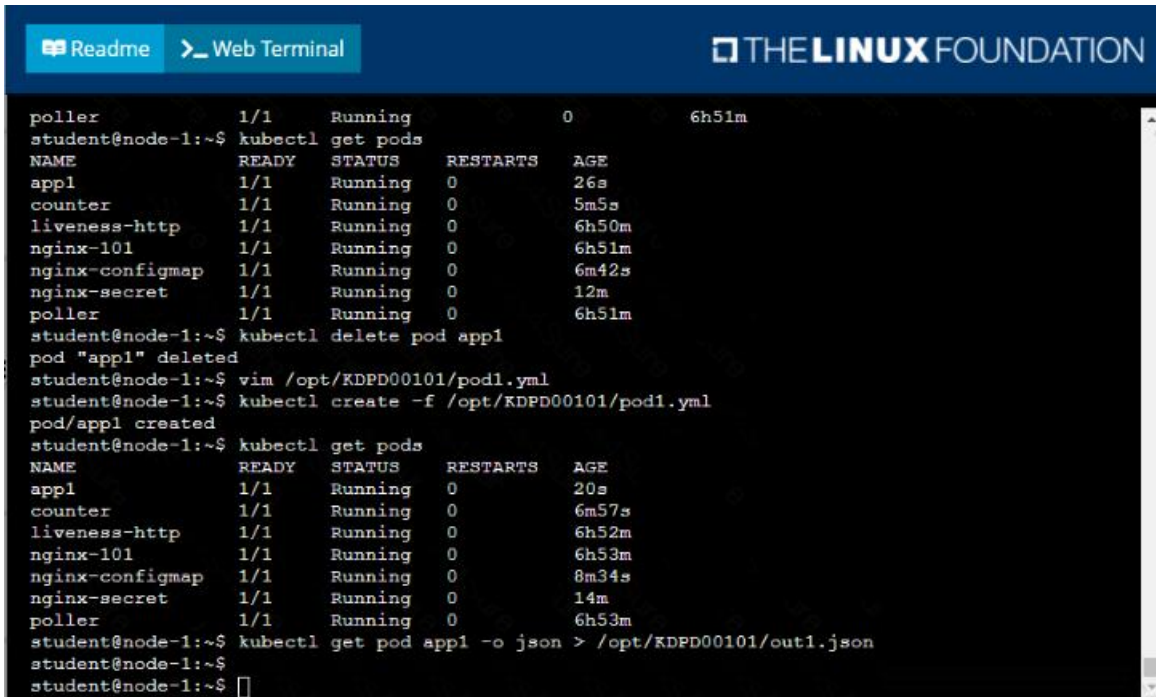
11,30 All



```
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           4m44s
liveness-http 1/1     Running             0           6h50m
nginx-101     1/1     Running             0           6h51m
nginx-configmap 1/1     Running             0           6m21s
nginx-secret   1/1     Running             0           11m
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
```

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 >
```

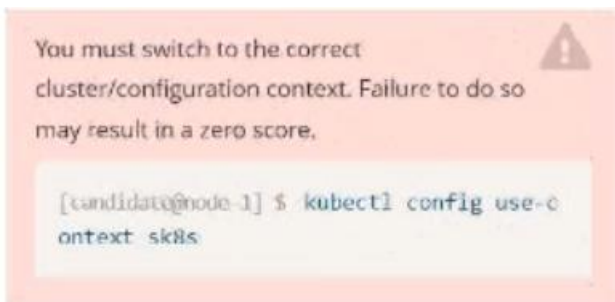


```

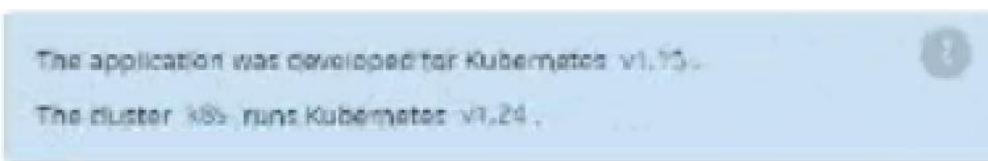
poller           1/1      Running           0           6h51m
student@node-1:~$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
appl          1/1     Running    0           26s
counter       1/1     Running    0           5m5s
liveness-http 1/1     Running    0           6h50m
nginx-101     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 appl
pod "appl" deleted
student@node-1:~$ vim /opt/KDPD00101/pod1.yml
student@node-1:~$ kubectl create -f /opt/KDPD00101/pod1.yml
pod/appl created
student@node-1:~$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
appl          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 appl -o json > /opt/KDPD00101/out1.json
student@node-1:~$
student@node-1:~$

```

28. Exhibit:



Task:



Answer:

See the solution below.

Explanation:

Solution:



```

[candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
[candidate@node-1:~$ vim ~/credible-mite/www.yaml

```

Text Description automatically generated



```

File Edit View Terminal Tabs Help
apiVersion: apps/v1
kind: Deployment
metadata:
  name: www-deployment
  namespace: cobra
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: "nginx:stable"
          ports:
            - containerPort: 80
          volumeMounts:
            - mountPath: /var/log/nginx
              name: logs
          env:
            - name: NGINX_ENTRYPOINT_QUIET_LOGS
              value: "1"
      volumes:
        - name: logs
          emptyDir: {}

```

Text Description automatically generated

```

File Edit View Terminal Tabs Help
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-fl7j              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:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ vim ~/credible-mite/www.yaml
candidate@node-1:~$ vim ~/credible-mite/www.yaml
candidate@node-1:~$ kubectl apply -f ~/credible-mite/www.yaml
deployment.apps/www-deployment created
candidate@node-1:~$ kubectl get pods -n cobra
NAME                                READY    STATUS              RESTARTS   AGE
www-deployment-d899c6b49-d6ccg      1/1     Running            0           6s
www-deployment-d899c6b49-f796l      0/1     ContainerCreating   0           6s
www-deployment-d899c6b49-ztfcw      0/1     ContainerCreating   0           6s
candidate@node-1:~$ kubectl get deploy -n cobra
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
www-deployment  3/3     3            3           11s
candidate@node-1:~$ kubectl get pods -n cobra
NAME                                READY    STATUS              RESTARTS   AGE
www-deployment-d899c6b49-d6ccg      1/1     Running            0           14s
www-deployment-d899c6b49-f796l      1/1     Running            0           14s
www-deployment-d899c6b49-ztfcw      1/1     Running            0           14s
candidate@node-1:~$

```

29. Exhibit:





### Context

A web application requires a specific version of redis to be used as a cache. Task

Create a pod with the following characteristics, and leave it running when complete:

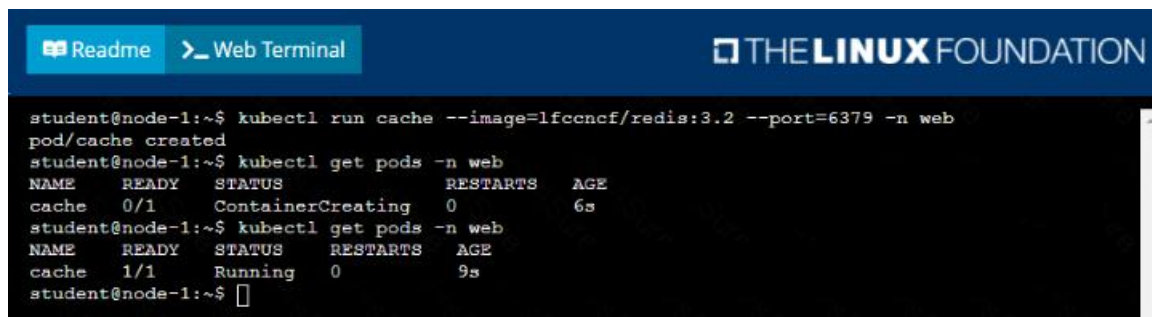
- The pod must run in the web namespace. The namespace has already been created
- The name of the pod should be cache
- Use the lfccncf/redis image with the 3.2 tag
- Expose port 6379

### Answer:

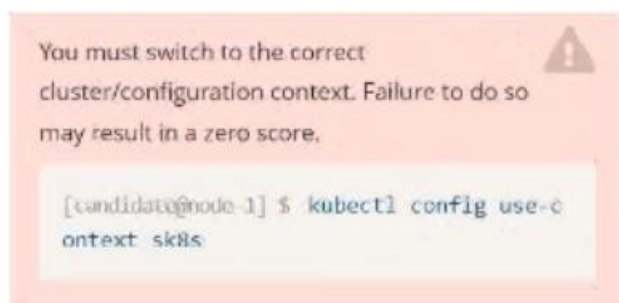
See the solution below.

Explanation:

Solution:



30. Exhibit:



Task:

Key3: value1

Add an environment variable named BEST\_VARIABLE consuming the value of the secret key3.

**Answer:**

See the solution below.

Explanation: Solution:

```
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl create secret generic app-secret -n default --from-literal=key3=value1
secret/app-secret created
candidate@node-1:~$ kubectl get secrets
NAME          TYPE      DATA   AGE
app-secret    Opaque    1       4s
candidate@node-1:~$ kubectl run nginx-secret -n default --image=nginx:stable --dry-run=client -o yaml > sec.yaml
candidate@node-1:~$ vim sec.yaml
```

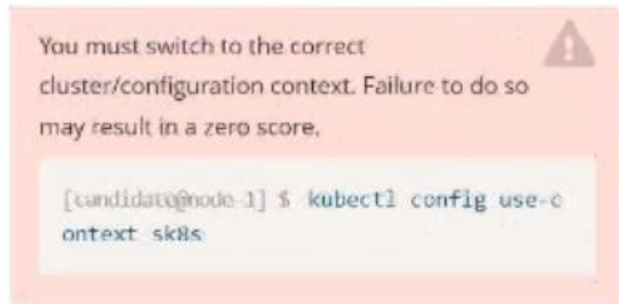
Text Description automatically generated

```
File Edit View Terminal Tabs Help
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx-secret
  name: nginx-secret
  namespace: default
spec:
  containers:
  - image: nginx:stable
    name: nginx-secret
    env:
    - name: BEST_VARIABLE
      valueFrom:
        secretKeyRef:
          name: app-secret
          key: key3
```

Text Description automatically generated

```
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ kubectl create secret generic app-secret -n default --from-literal=key3=value1
secret/app-secret created
candidate@node-1:~$ kubectl get secrets
NAME          TYPE      DATA   AGE
app-secret    Opaque    1       4s
candidate@node-1:~$ kubectl run nginx-secret -n default --image=nginx:stable --dry-run=client -o yaml > sec.yaml
candidate@node-1:~$ vim sec.yaml
candidate@node-1:~$ kubectl create -f sec.yaml
pod/nginx-secret created
candidate@node-1:~$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
nginx-secret  1/1     Running   0          7s
candidate@node-1:~$
```

31. Exhibit:



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:**



See the solution below.

Explanation:

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:~$
candidate@node-1:~$
```

Text Description automatically generated

```

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"

```

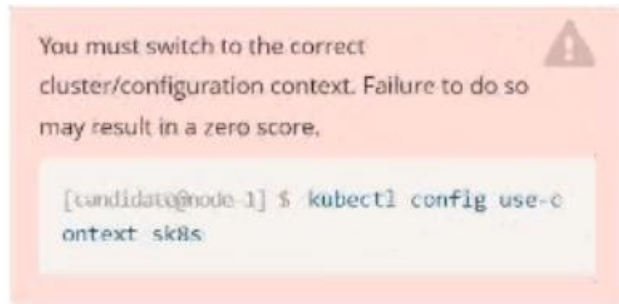
Text Description automatically generated

```

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 > dep.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:~$ 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:~$

```

32. Exhibit:



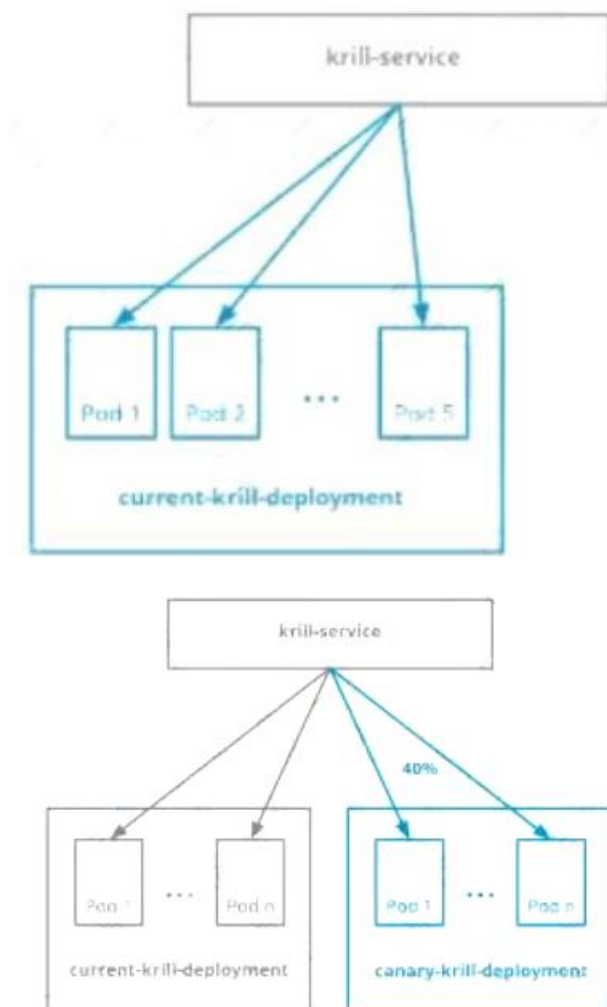
## Context

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







**Answer:**

See the solution below.

**Explanation:**

**Solution:**

```
candidate@node-1:~/humane-storks$ kubectl scale deploy canary-krill-deployment --replicas 4 -n goshawk
deployment.apps/canary-krill-deployment scaled
candidate@node-1:~/humane-storks$ 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-storks$ wget https://k8s.io/examples/
```

Text Description automatically generated

```
File Edit View Terminal Tabs Help
candidate@node-1:~/humane-storks$ 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]

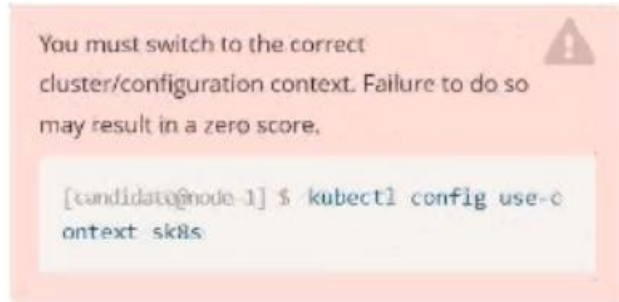
candidate@node-1:~/humane-storks$ vim quota-pod.yaml
```

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

candidate@node-1:~/humane-storks$ vim quota-pod.yaml
candidate@node-1:~/humane-storks$ kubectl create -f quota-pod.yaml
resourcequota/pod-demo created
candidate@node-1:~/humane-storks$ kubectl get quota -n go
No resources found in go namespace.
candidate@node-1:~/humane-storks$ kubectl get quota -n goshawk
NAME      AGE  REQUEST  LIMIT
pod-demo  19s  pods: 9/10
candidate@node-1:~/humane-storks$ 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$
```

33. Exhibit:





Task:

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

**Answer:**

See the solution below.

Explanation: Solution:

Text Description automatically generated

```
File Edit View Terminal Tabs Help
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:~$
```