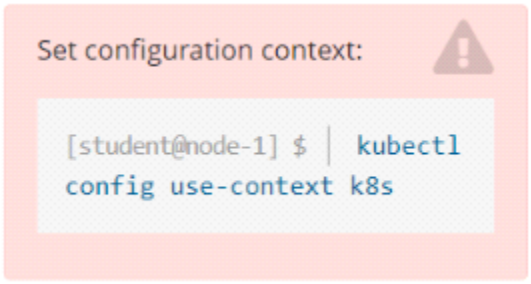
CKAD Exam Questions

Q1



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 Ifccncf/redis image with the 3.2 tag

* Expose port 6379

A Solution:

B Solution:

Hide Answer

Suggested Answer: *A*

Q2

Context

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

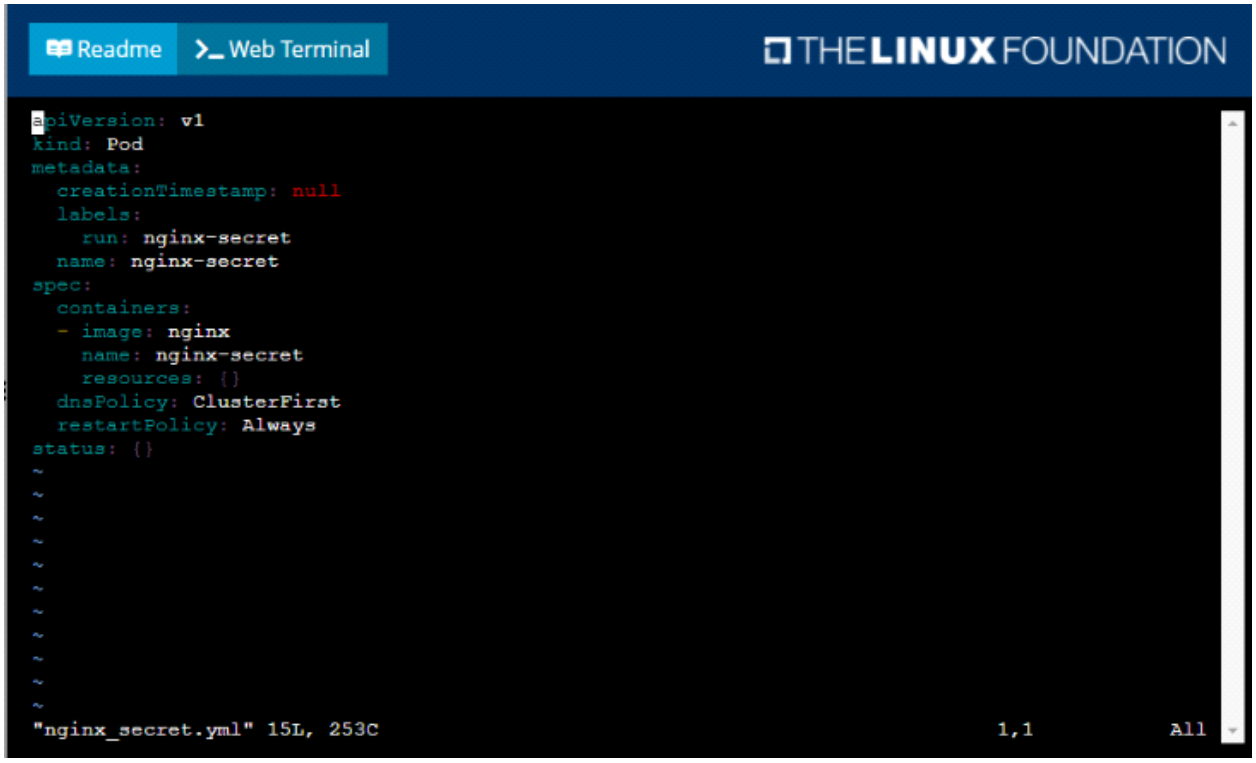
Task

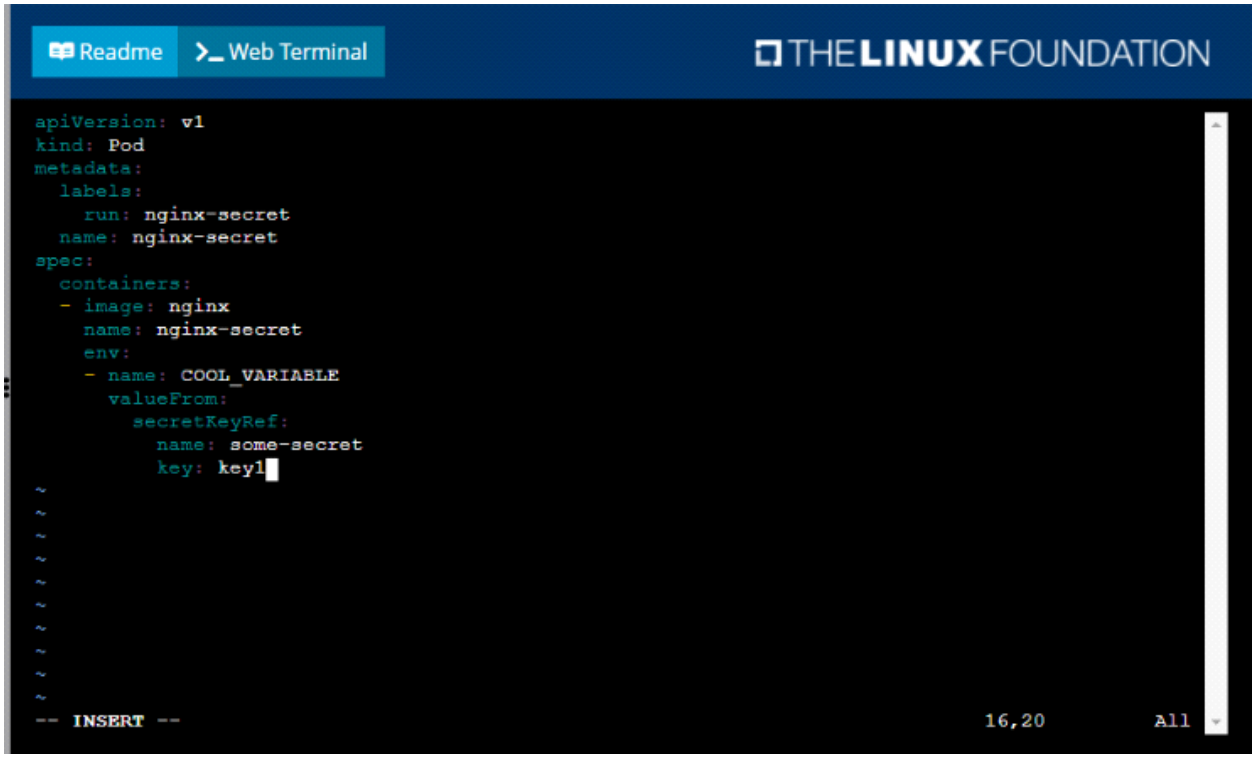
* Create a secret named another-secret with a key/value pair; key1/value4

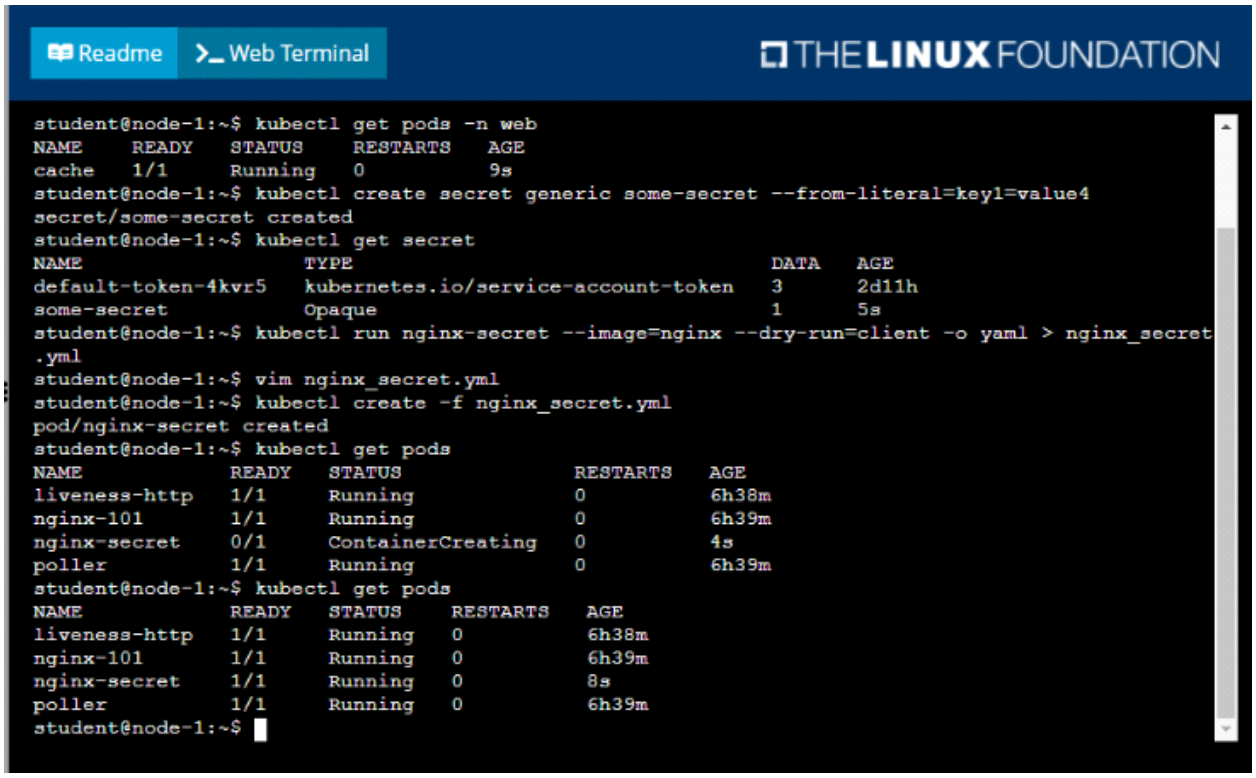
* Start an nginx pod named nginx-secret using container image nginx, and add an environment variable exposing the value of the secret key key 1, using COOL_VARIABLE as the name for the environment variable inside the pod

ASolution:

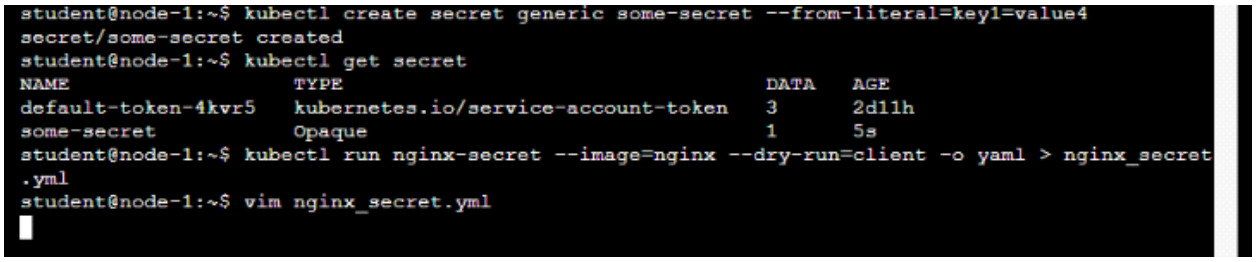
```
student@node-1:~$ kubectl create secret generic some-secret --from-literal=key1=value4
secret/some-secret created
student@node-1:~$ kubectl get secret
NAME                  TYPE                                  DATA   AGE
default-token-4kvr5   kubernetes.io/service-account-token   3      2d11h
some-secret           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
```

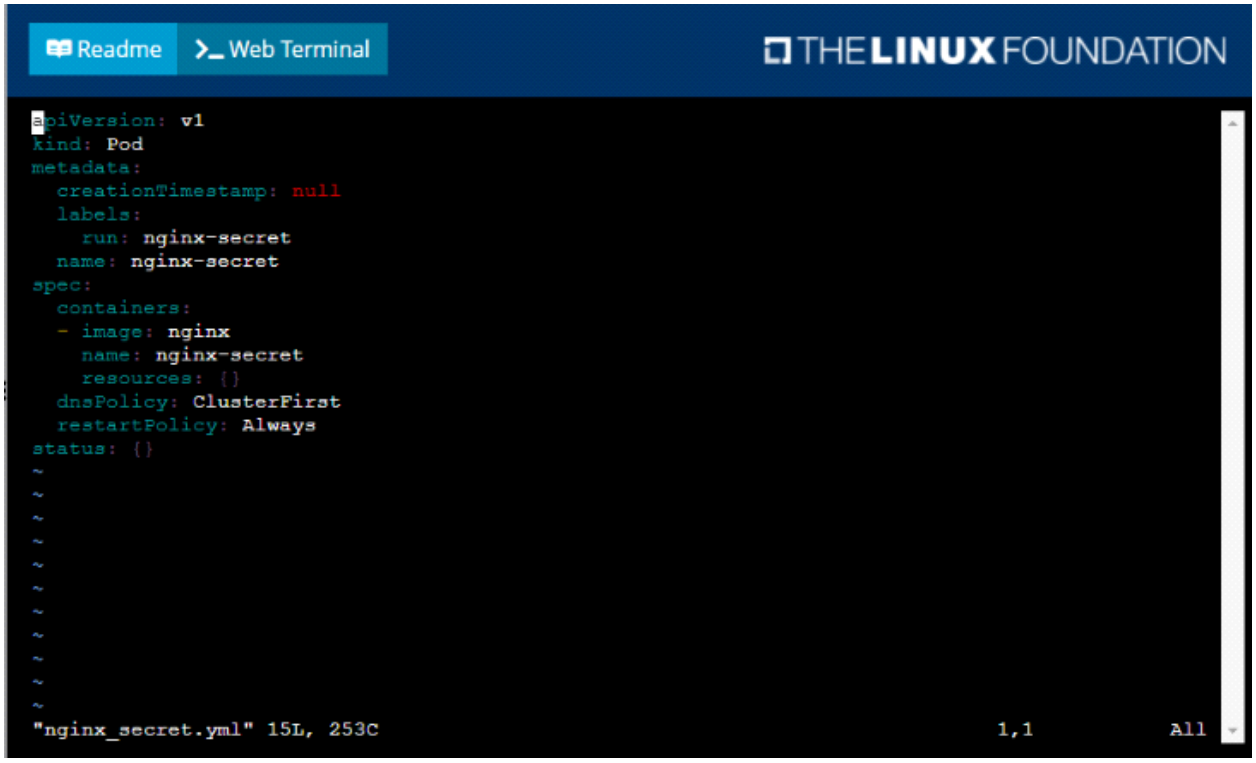

Readme
Web Terminal
THE LINUX FOUNDATION
apiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
labels:
run: nginx-secret
name: nginx-secret
spec:
containers:
- image: nginx
name: nginx-secret
resources: {}
dnsPolicy: ClusterFirst
restartPolicy: Always
status: {}
"nginx_secret.yml" 15L, 253C
1,1
All

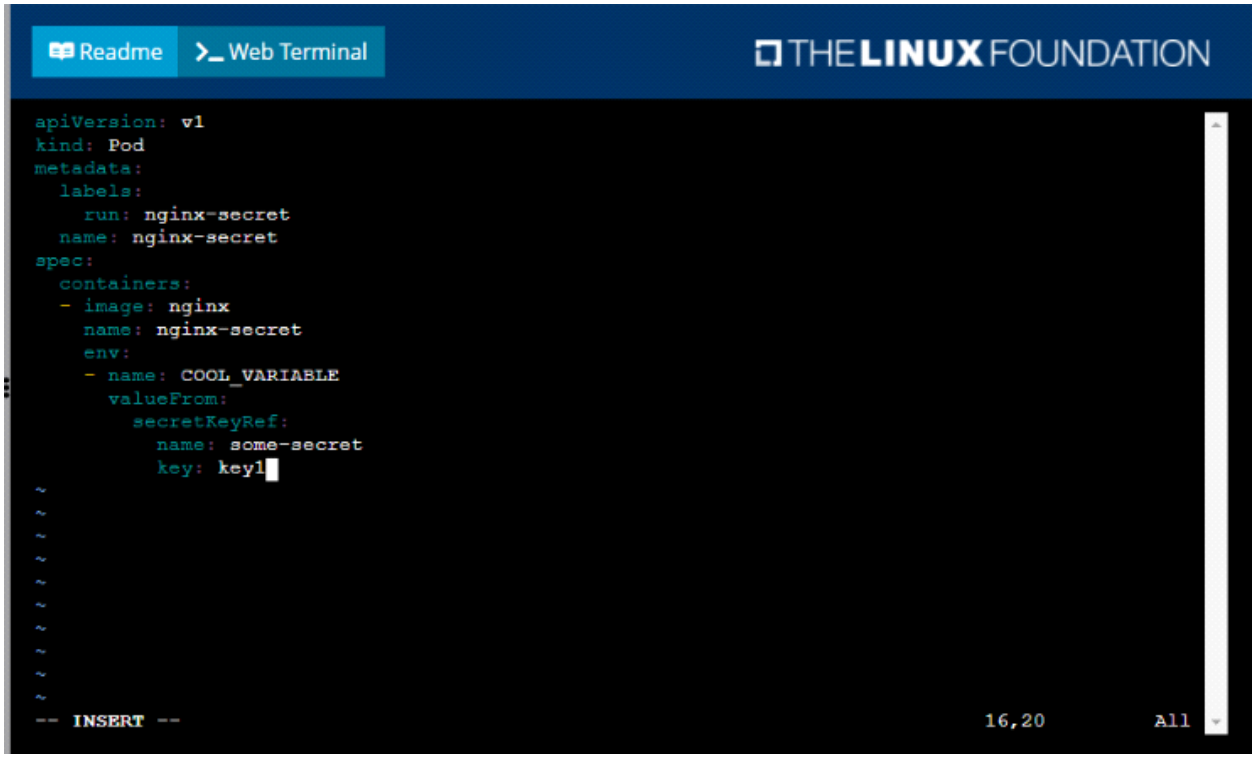

Readme
Web Terminal
THE LINUX FOUNDATION
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx-secret
name: nginx-secret
spec:
containers:
- image: nginx
name: nginx-secret
env:
- name: COOL_VARIABLE
valueFrom:
secretKeyRef:
name: some-secret
key: key1
-- INSERT --
16,20
All

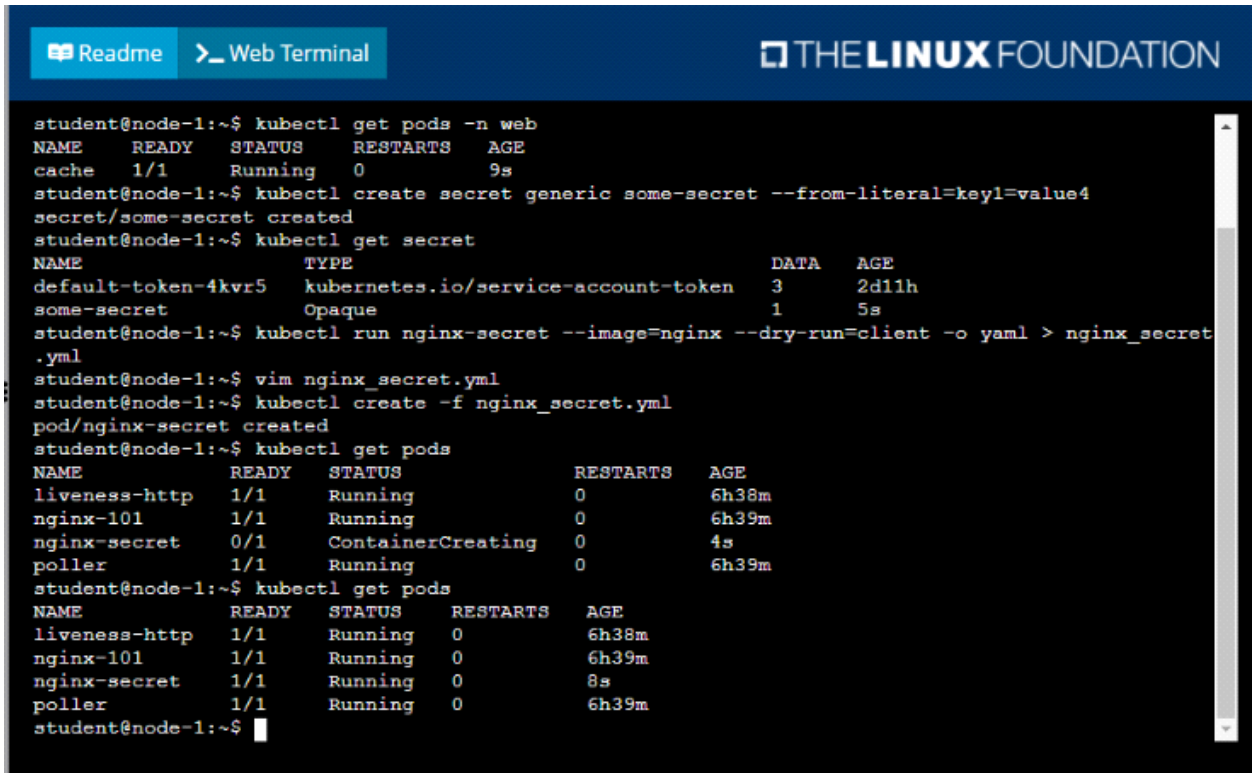

Readme
Web Terminal
THE LINUX FOUNDATION
student@node-1:~$ kubectl get pods -n web
NAME READY STATUS RESTARTS AGE
cache 1/1 Running 0 9s
student@node-1:~$ kubectl create secret generic some-secret --from-literal=key1=value4
secret/some-secret created
student@node-1:~$ kubectl get secret
NAME TYPE DATA 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
student@node-1:~$ kubectl create -f nginx_secret.yml
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:~$

B 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



Hide Answer

Suggested Answer: *B*

Q3

Task

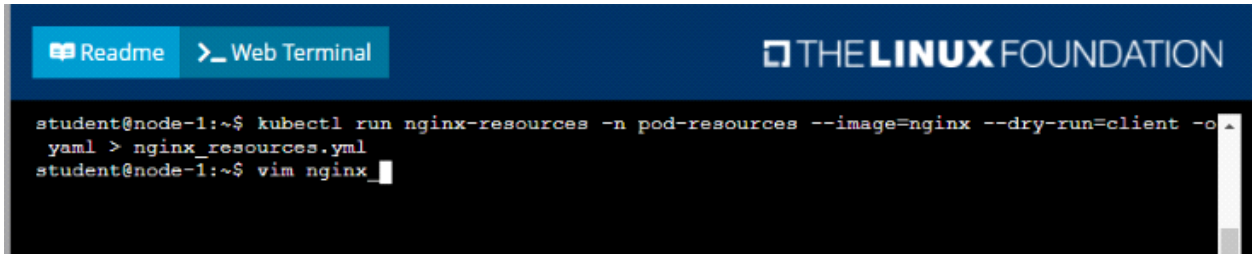
You are required to create a pod that requests a certain amount of CPU and memory, so it gets scheduled to-a node that has those resources available.

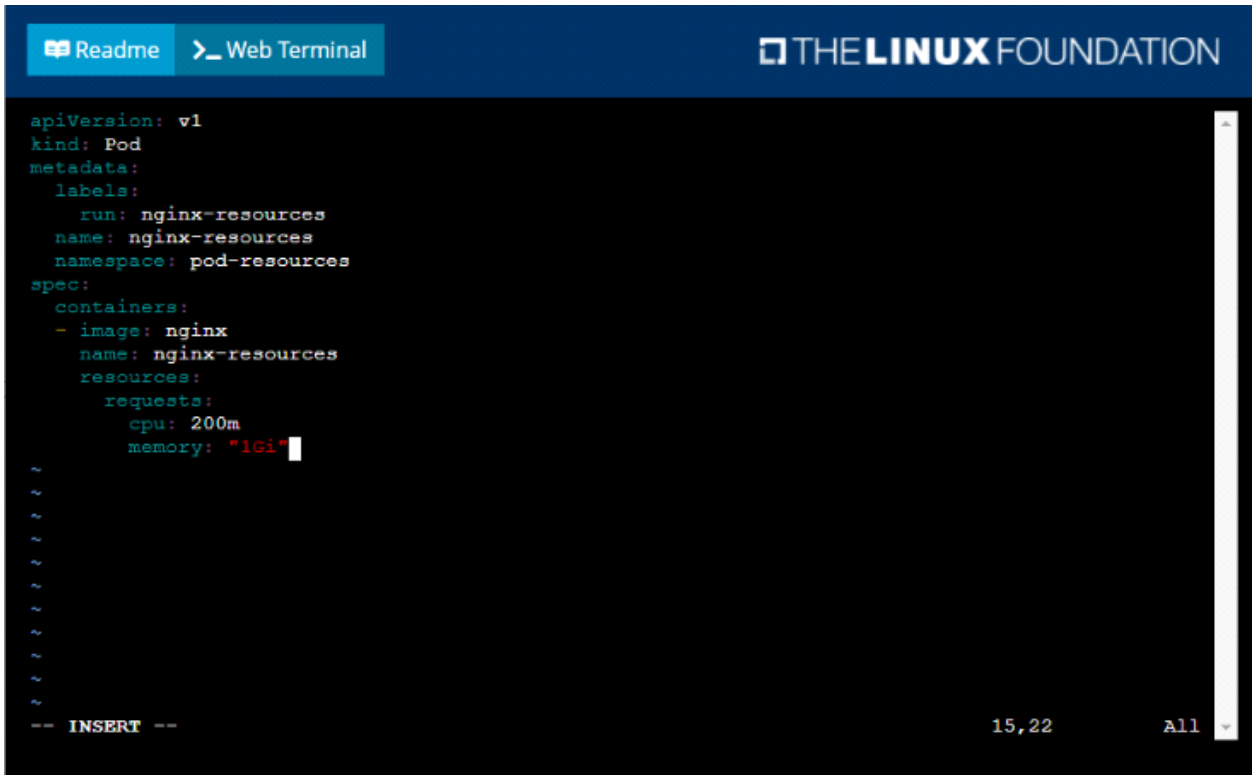
* Create a pod named nginx-resources in the pod-resources namespace that requests a minimum of 200m CPU and 1Gi memory for its container

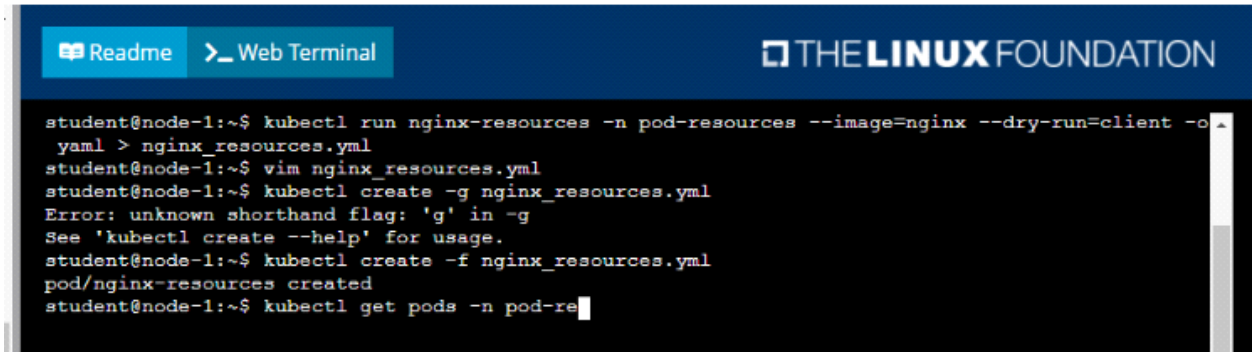
* The pod should use the nginx image

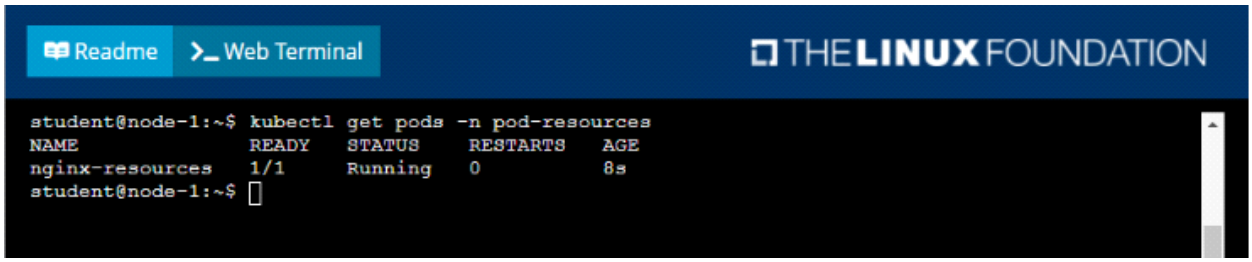
* The pod-resources namespace has already been created

Solution:

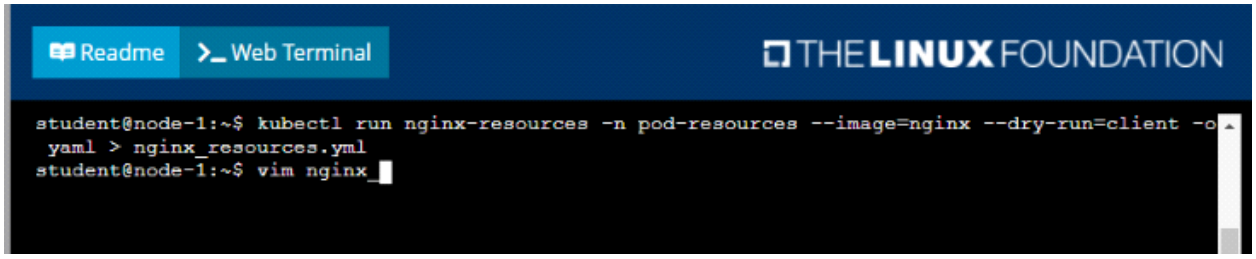



Readme
Web Terminal
THE LINUX FOUNDATION
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx-resources
name: nginx-resources
namespace: pod-resources
spec:
containers:
- image: nginx
name: nginx-resources
resources:
requests:
cpu: 200m
memory: "1Gi"
-- INSERT --
15,22
All

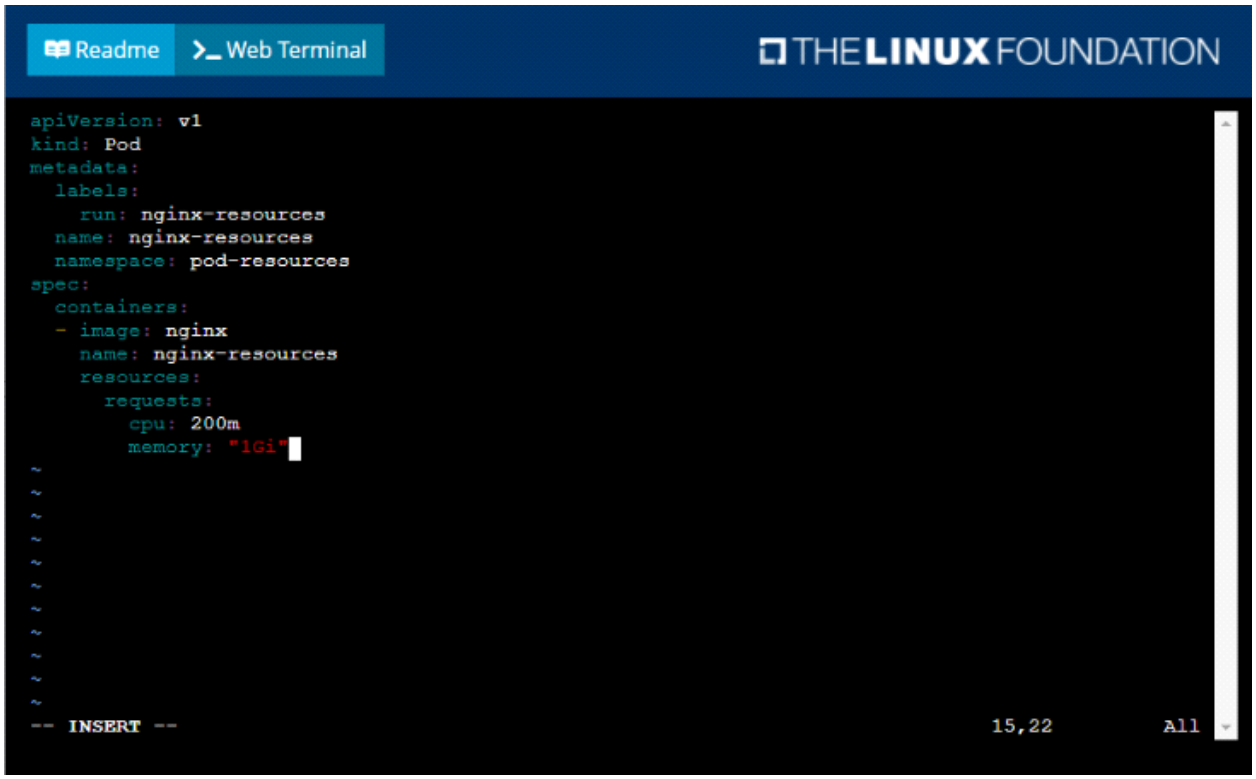

Readme
Web Terminal
THE LINUX FOUNDATION
student@node-1:~$ kubectl run nginx-resources -n pod-resources --image=nginx --dry-run=client -o
yaml > nginx_resources.yml
student@node-1:~$ vim nginx_resources.yml
student@node-1:~$ kubectl create -g nginx_resources.yml
Error: unknown shorthand flag: 'g' in -g
See 'kubectl create --help' for usage.
student@node-1:~$ kubectl create -f nginx_resources.yml
pod/nginx-resources created
student@node-1:~$ kubectl get pods -n pod-re

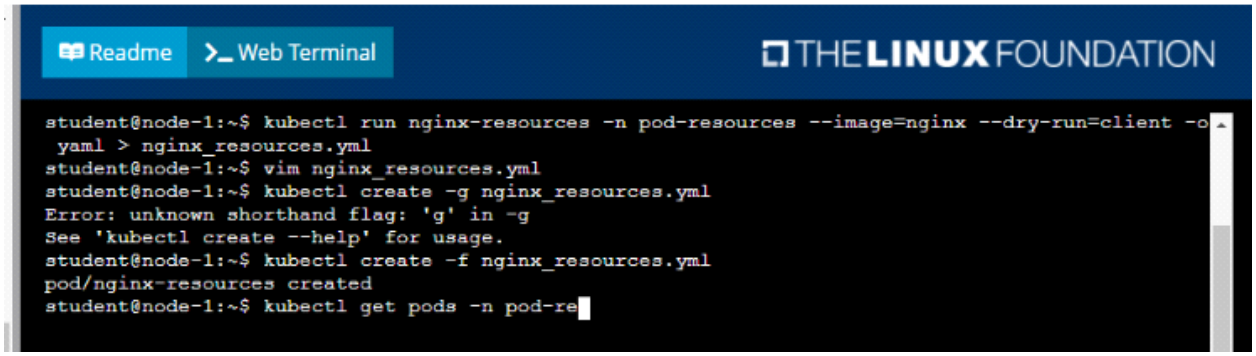

Readme
Web Terminal
THE LINUX FOUNDATION
student@node-1:~$ kubectl get pods -n pod-resources
NAME READY STATUS RESTARTS AGE
nginx-resources 1/1 Running 0 8s
student@node-1:~$

B Solution:


Readme
Web Terminal
THE LINUX FOUNDATION
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


Readme
Web Terminal
THE LINUX FOUNDATION
apiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
labels:
run: nginx-resources
name: nginx-resources
namespace: pod-resources
spec:
containers:
- image: nginx
name: nginx-resources
resources: {}
dnsPolicy: ClusterFirst
restartPolicy: Always
status: {}
"nginx_resources.yml" 16L, 289C
1,1
All





Hide Solution

Correct Answer: *A*

Q4

Set configuration context:

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

Context

You are tasked to create a ConfigMap and consume the ConfigMap in a pod using a volume mount.

Task

Please complete the following:

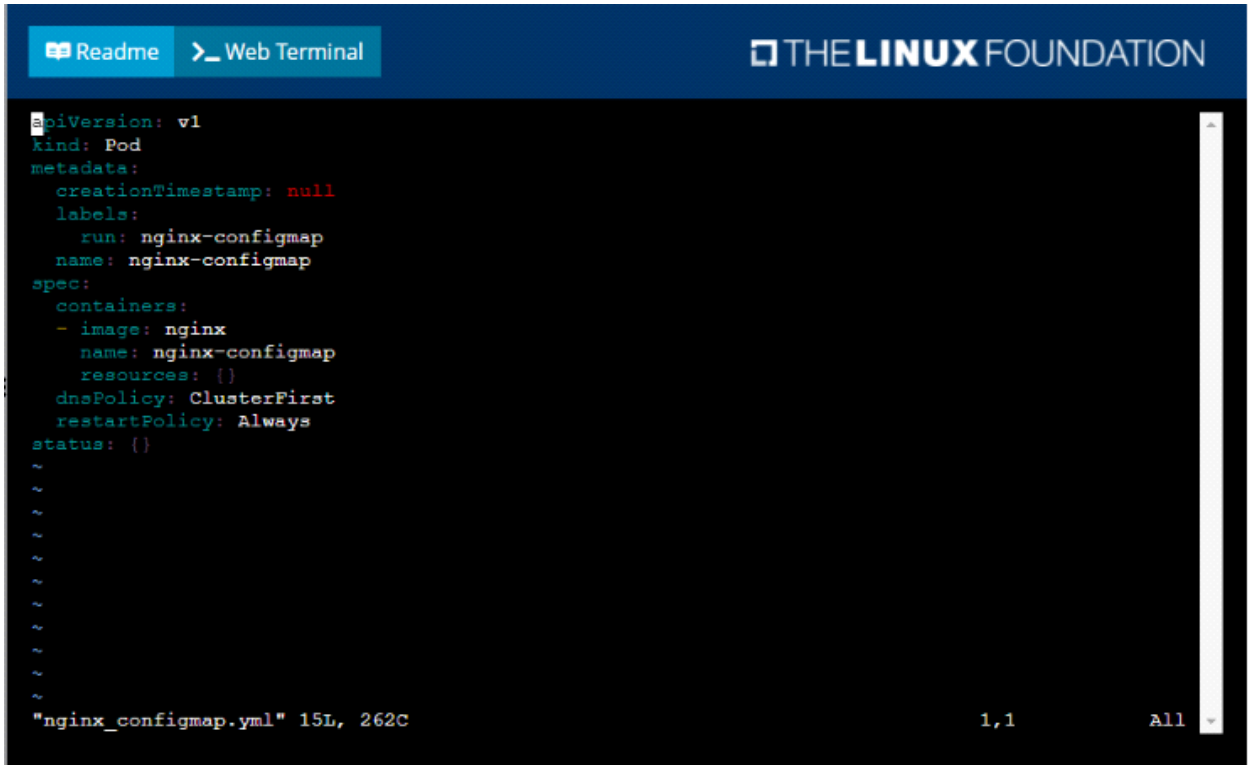
* Create a ConfigMap named another-config containing the key/value pair: key4/value3

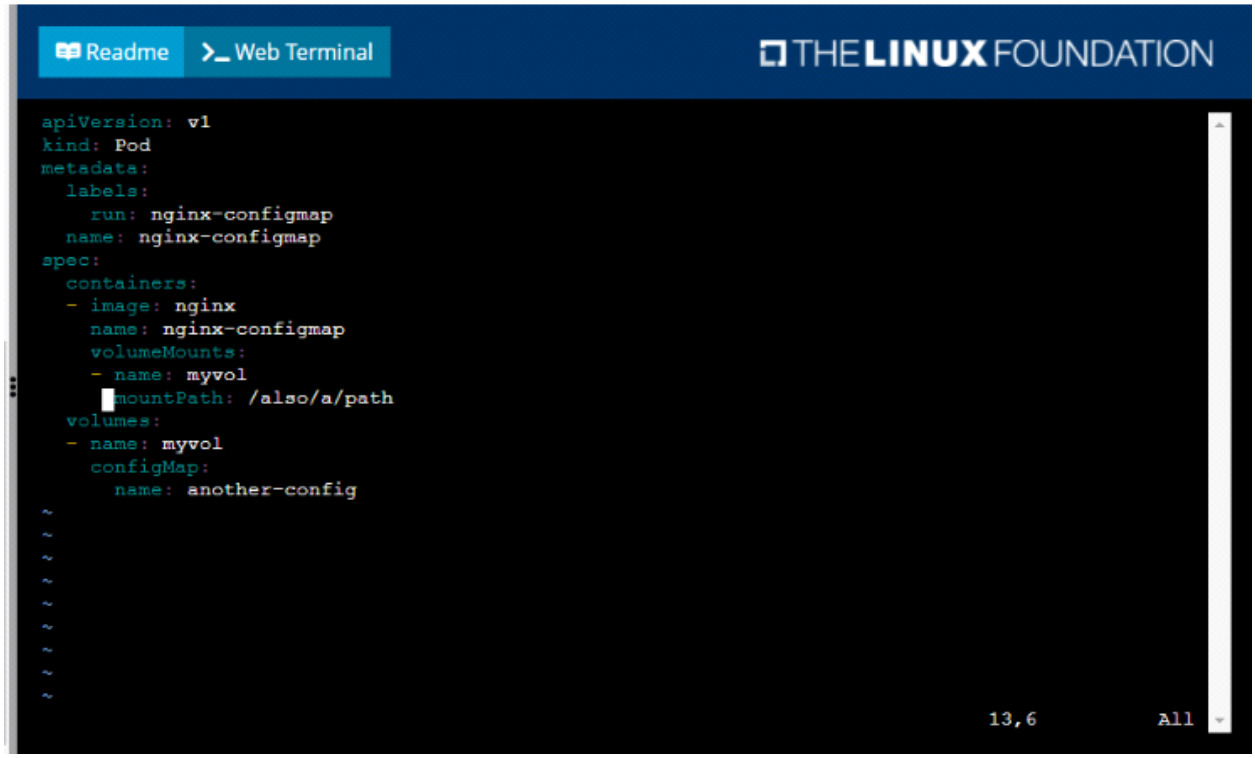
* start a pod named nginx-configmap containing a single container using the

nginx image, and mount the key you just created into the pod under directory /also/a/path

ASolution:

```
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 > ngin_conf
igmap.yml
student@node-1:~$ vim ngin_configmap.yml ^C
student@node-1:~$ mv ngin_configmap.yml nginx_configmap.yml
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.yml" 15L, 262C
1,1
All

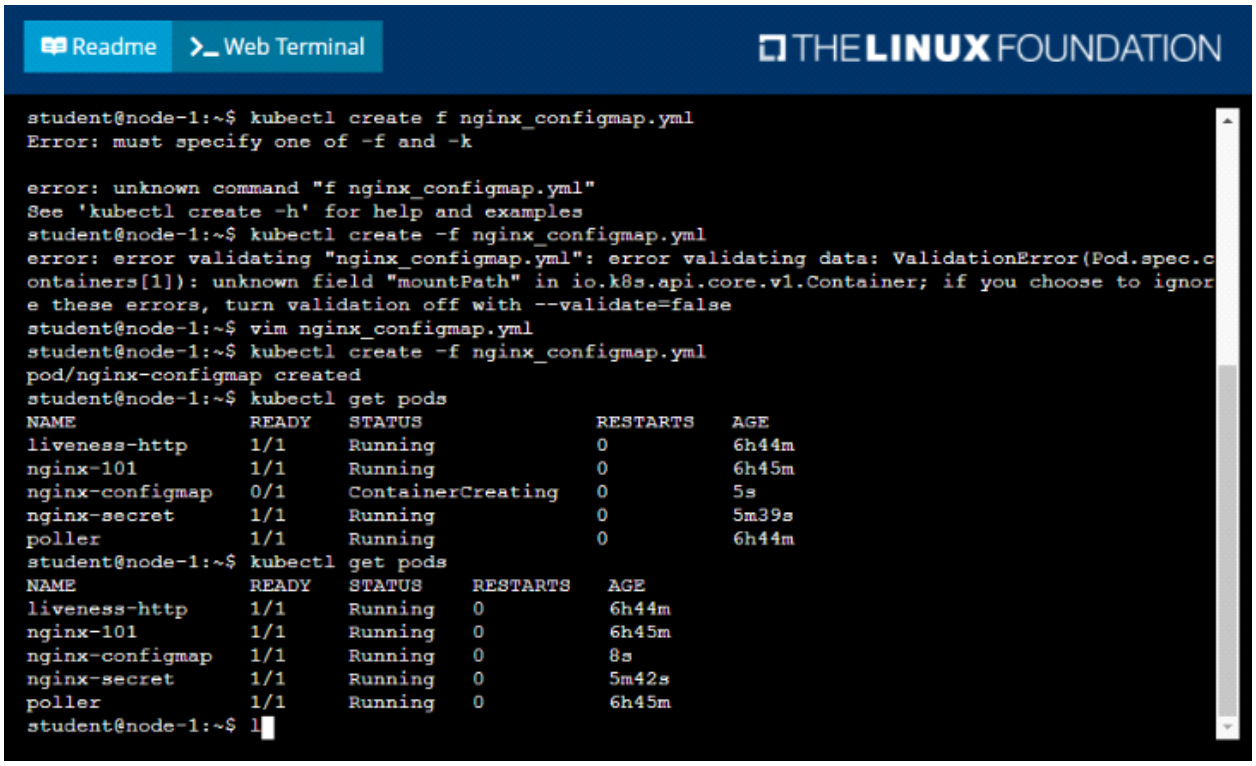

Readme
Web Terminal
THE LINUX FOUNDATION
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx-configmap
name: nginx-configmap
spec:
containers:
- image: nginx
name: nginx-configmap
volumeMounts:
- name: myvol
mountPath: /also/a/path
volumes:
- name: myvol
configMap:
name: another-config
13,6
All

```
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 > ngin_conf
igmap.yml
student@node-1:~$ vim ngin_configmap.yml ^C
student@node-1:~$ mv ngin_configmap.yml nginx_configmap.yml
student@node-1:~$ vim nginx_configmap.yml
student@node-1:~$
```

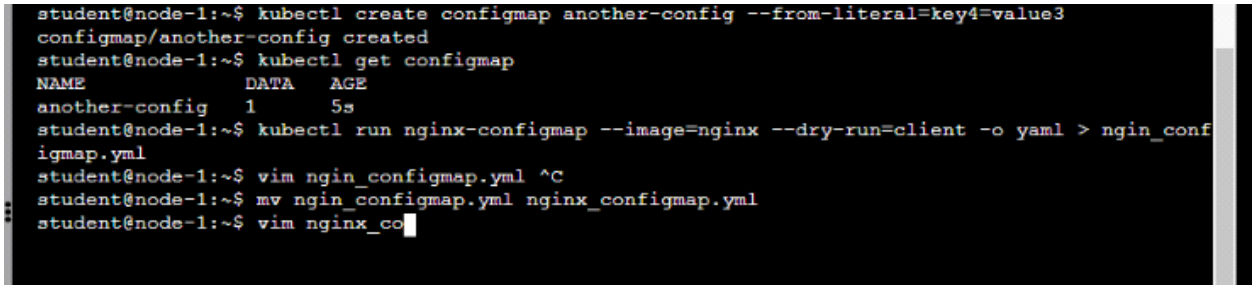
Readme | Web Terminal

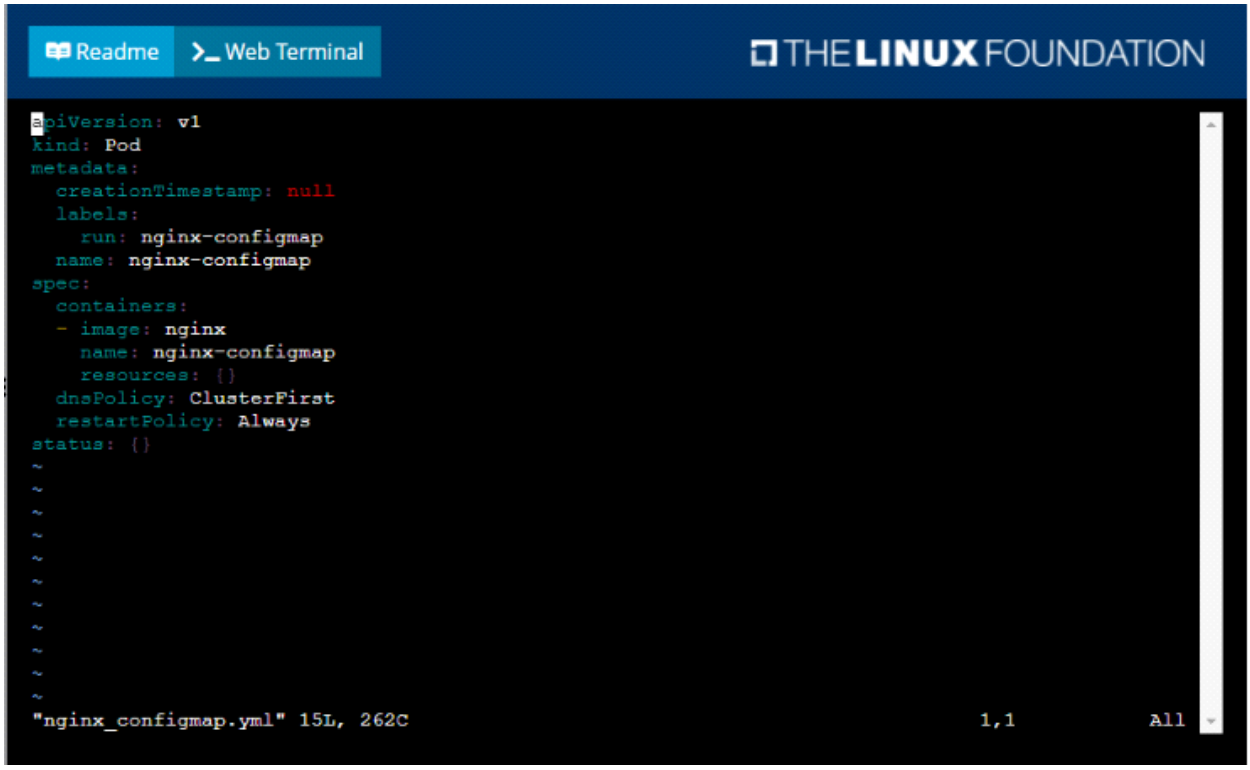
THE LINUX FOUNDATION

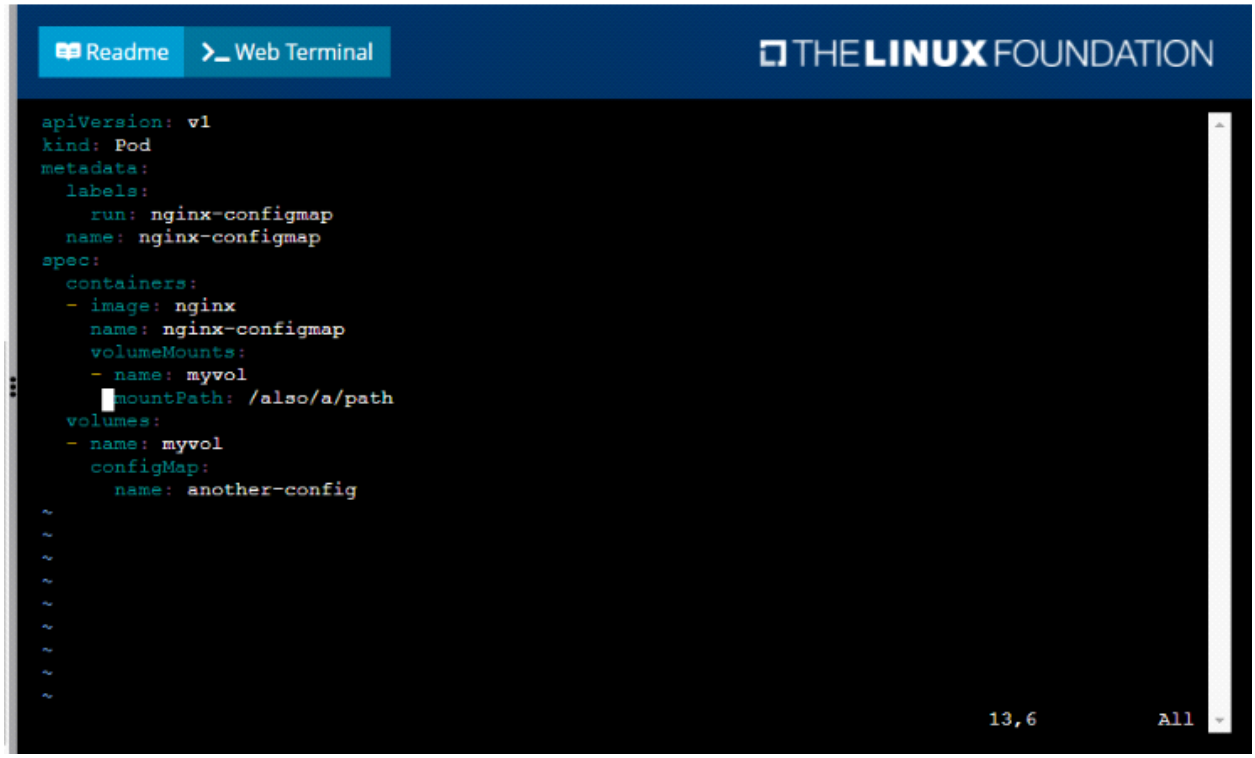
```
student@node-1:~$ kubectl create f nginx_configmap.yml
Error: must specify one of -f and -k

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.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.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:~$ l
```


Readme
Web Terminal
THE LINUX FOUNDATION
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.containers[1]): unknown field "mountPath" in io.k8s.api.core.v1.Container; if you choose to ignore 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:~$ l

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 > ngin_configmap.yml
student@node-1:~$ vim ngin_configmap.yml ^C
student@node-1:~$ mv ngin_configmap.yml nginx_configmap.yml
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.yml" 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 > ngin_conf
igmap.yml
student@node-1:~$ vim ngin_configmap.yml ^C
student@node-1:~$ mv ngin_configmap.yml nginx_configmap.yml
student@node-1:~$ vim nginx_configmap.yml
student@node-1:~$
```

Readme | Web Terminal — THE LINUX FOUNDATION

```
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.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.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:~$ l
```

Correct Answer: *A*

Q5

Set configuration context:

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

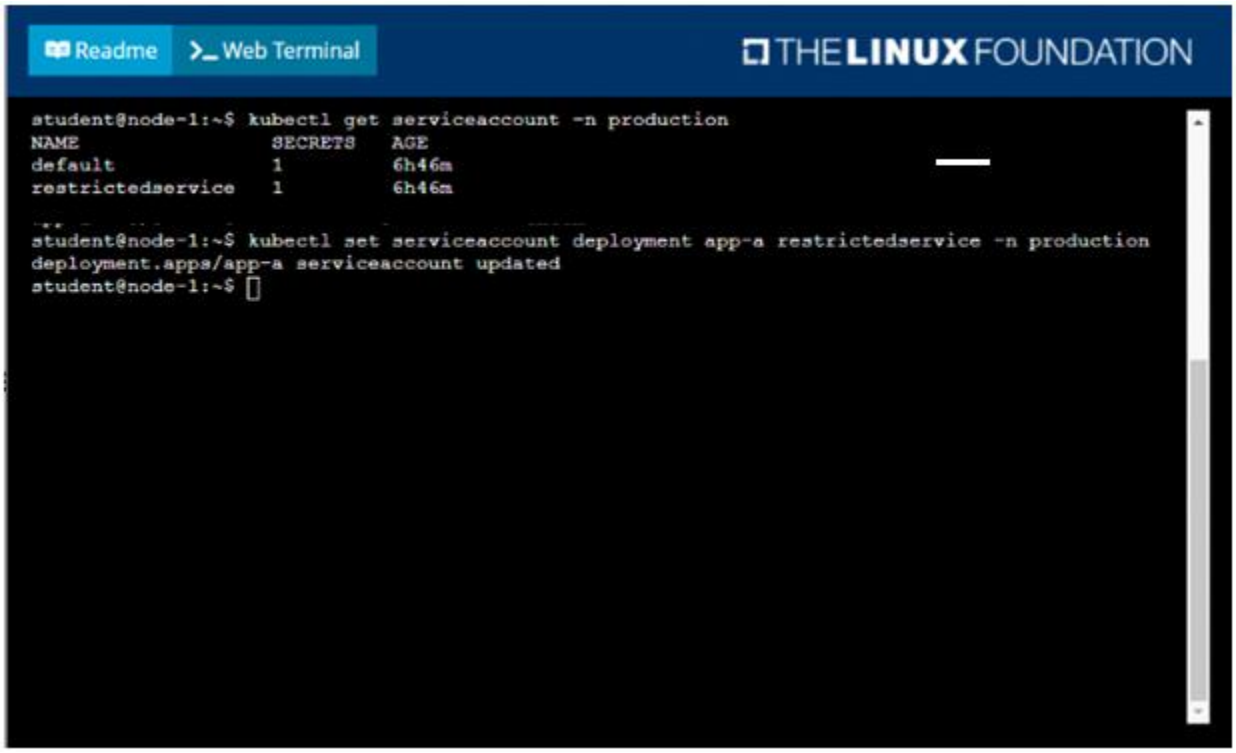
Context

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

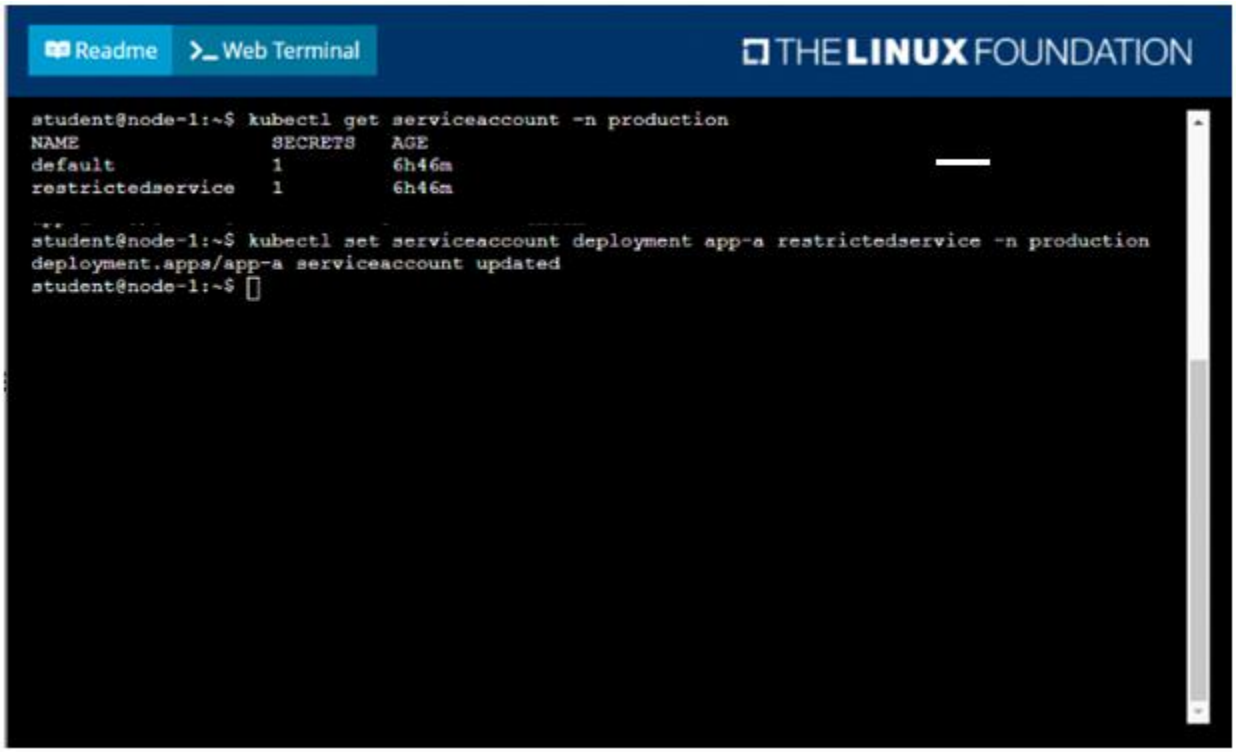
Task

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

Solution:



Solution:



Hide Solution Discuss 0

Correct Answer: *A*

Q6

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

ASolution:

```
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 singleContainer. TheperiodSecondsfield specifies that the kubelet should perform a liveness probe every 5 seconds. TheinitialDelaySecondsfield tells the kubelet that it should wait 5 seconds before performing the

first probe. To perform a probe, the kubelet executes the commandcat /tmp/healthyin 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/healthyfile. So during the first 30 seconds, the commandcat /tmp/healthyreturns a success code. After 30 seconds,cat /tmp/healthyreturns a failure code.
Create the Pod:
kubectl apply -f https://k8s.io/examples/pods/probe/exec-liveness.yaml
Within 30 seconds, view the Pod events:
kubectl describe pod liveness-exec
The output indicates that no liveness probes have failed yet:
FirstSeen LastSeen Count From SubobjectPath Type Reason Message
--------- -------- ----- ---- ------------ -------- ------ -------
24s 24s 1 {default-scheduler } Normal Scheduled Successfully assigned liveness-exec to worker0
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image 'k8s.gcr.io/busybox'
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image 'k8s.gcr.io/busybox'
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Created Created container with docker id 86849c15382e; Security:[seccomp=unconfined]
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Started Started container with docker id 86849c15382e
After 35 seconds, view the Pod events again:
kubectl describe pod liveness-exec
At the bottom of the output, there are messages indicating that the liveness probes have failed, and the containers have been killed and recreated.
FirstSeen LastSeen Count From SubobjectPath Type Reason Message
--------- -------- ----- ---- ------------ -------- ------ -------
37s 37s 1 {default-scheduler } Normal Scheduled Successfully assigned liveness-exec to worker0
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image 'k8s.gcr.io/busybox'
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully
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 thatRESTARTShas been incremented:
NAME READY STATUS RESTARTS AGE
liveness-exec 1/1 Running 1 1m

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 singleContainer. TheperiodSecondsfield specifies that the kubelet should perform a liveness probe every 5 seconds. TheinitialDelaySecondsfield tells the kubelet that it should wait 5 seconds before performing the first probe. To perform a probe, the kubelet executes the commandcat /tmp/healthyin 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/healthyfile. So during the first 30 seconds, the commandcat /tmp/healthyreturns a success code. After 30 seconds,cat /tmp/healthyreturns a failure code.
Create the Pod:
kubectl apply -f https://k8s.io/examples/pods/probe/exec-liveness.yaml
Within 30 seconds, view the Pod events:
kubectl describe pod liveness-exec
The output indicates that no liveness probes have failed yet:
FirstSeen LastSeen Count From SubobjectPath Type Reason Message
--------- -------- ----- ---- ------------- -------- ------ -------
24s 24s 1 {default-scheduler } Normal Scheduled Successfully assigned liveness-exec to worker0
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image 'k8s.gcr.io/busybox'
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image 'k8s.gcr.io/busybox'

23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Created Created container with docker id 86849c15382e; Security:[seccomp=unconfined]
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Started Started container with docker id 86849c15382e
After 35 seconds, view the Pod events again:
kubectl describe pod liveness-exec
At the bottom of the output, there are messages indicating that the liveness probes have failed, and the containers have been killed and recreated.
FirstSeen LastSeen Count From SubobjectPath Type Reason Message
--------- -------- ----- ---- ------------- -------- ------ -------
37s 37s 1 {default-scheduler } Normal Scheduled Successfully assigned liveness-exec to worker0
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image 'k8s.gcr.io/busybox'
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image 'k8s.gcr.io/busybox'
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Created Created container with docker id 86849c15382e; Security:[seccomp=unconfined]
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Started Started container with docker id 86849c15382e
2s 2s 1 {kubelet worker0} spec.containers{liveness} Warning Unhealthy Liveness probe failed: cat: can't open '/tmp/healthy': No such file or directory
Wait another 30 seconds, and verify that the container has been restarted:
kubectl get pod liveness-exec
The output shows thatRESTARTShas been incremented:
NAME READY STATUS RESTARTS AGE
liveness-exec 1/1 Running 1 1m

Hide Answer
Suggested Answer: *B*

Q7

Context

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

Task

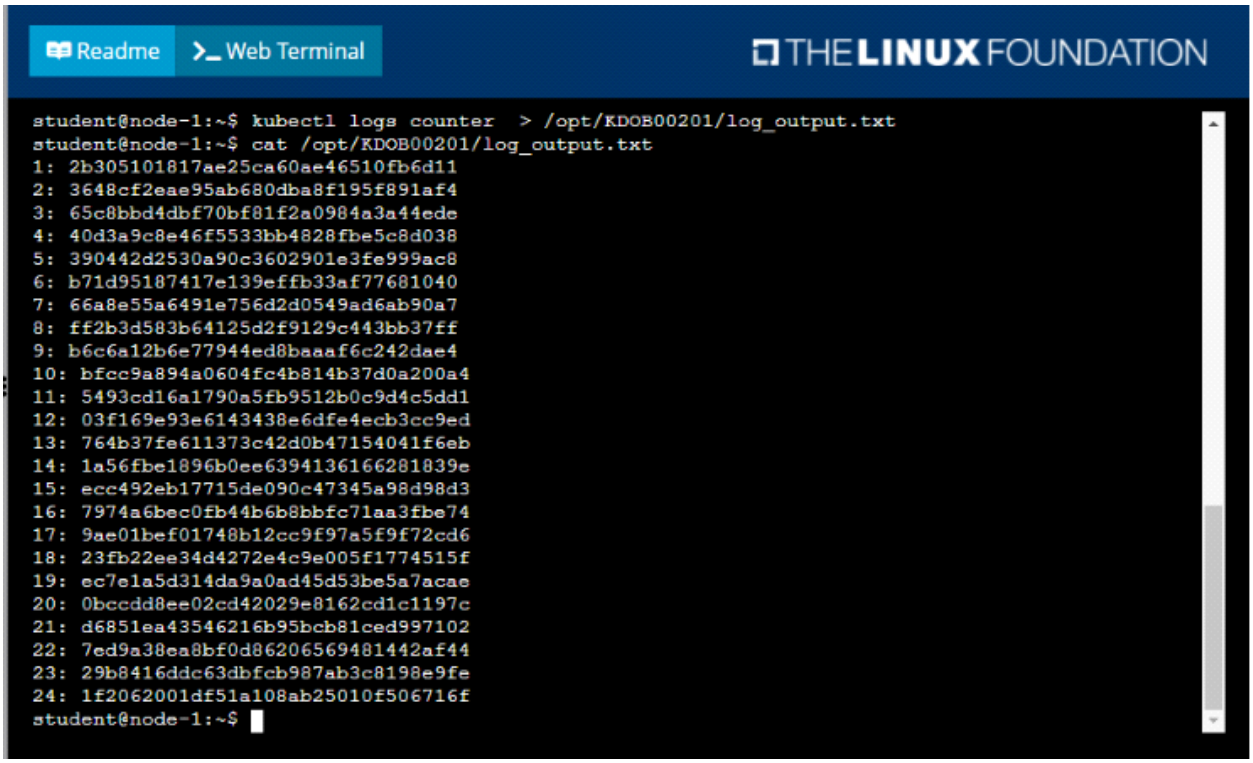
Please complete the following;

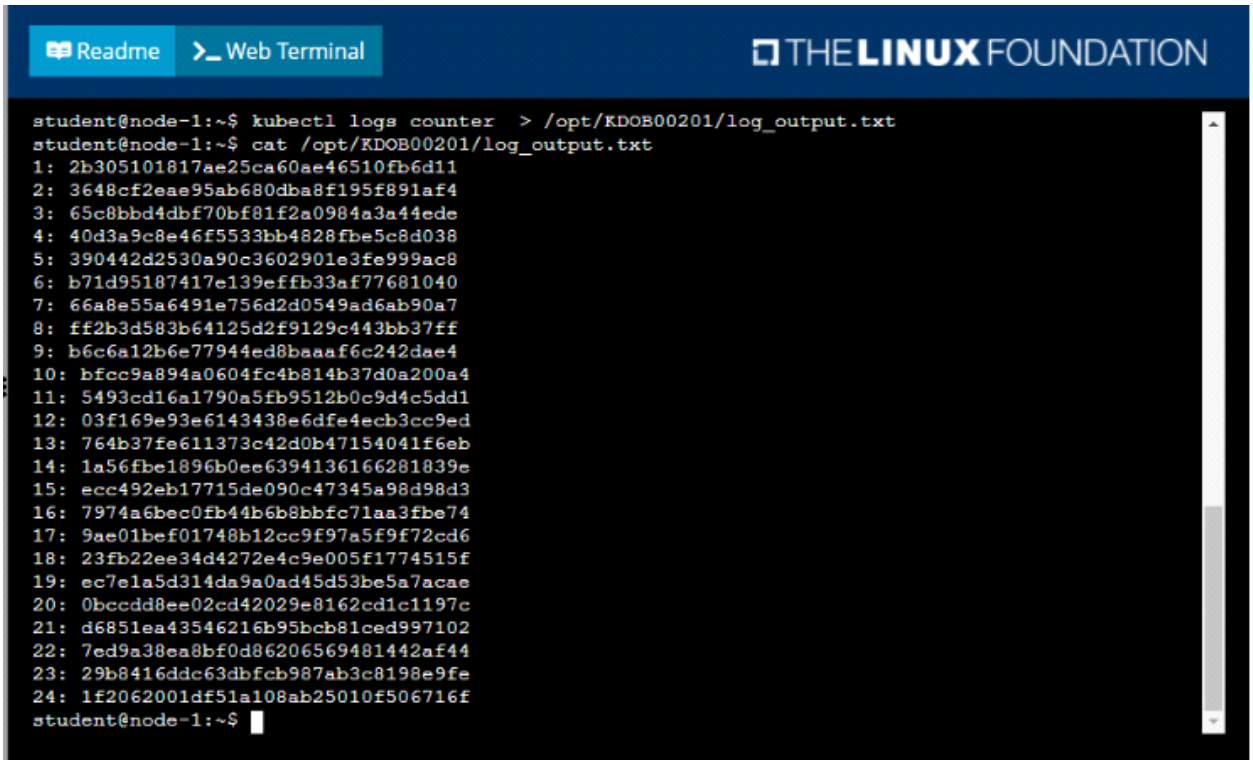
* Deploy the counter pod to the cluster using the provided YAMLspec file at /opt/KDOB00201/counter.yaml

* Retrieve all currently available application logs from the running pod and store them in the file /opt/KDOB0020l/log_Output.txt, which has already been created

A Solution:

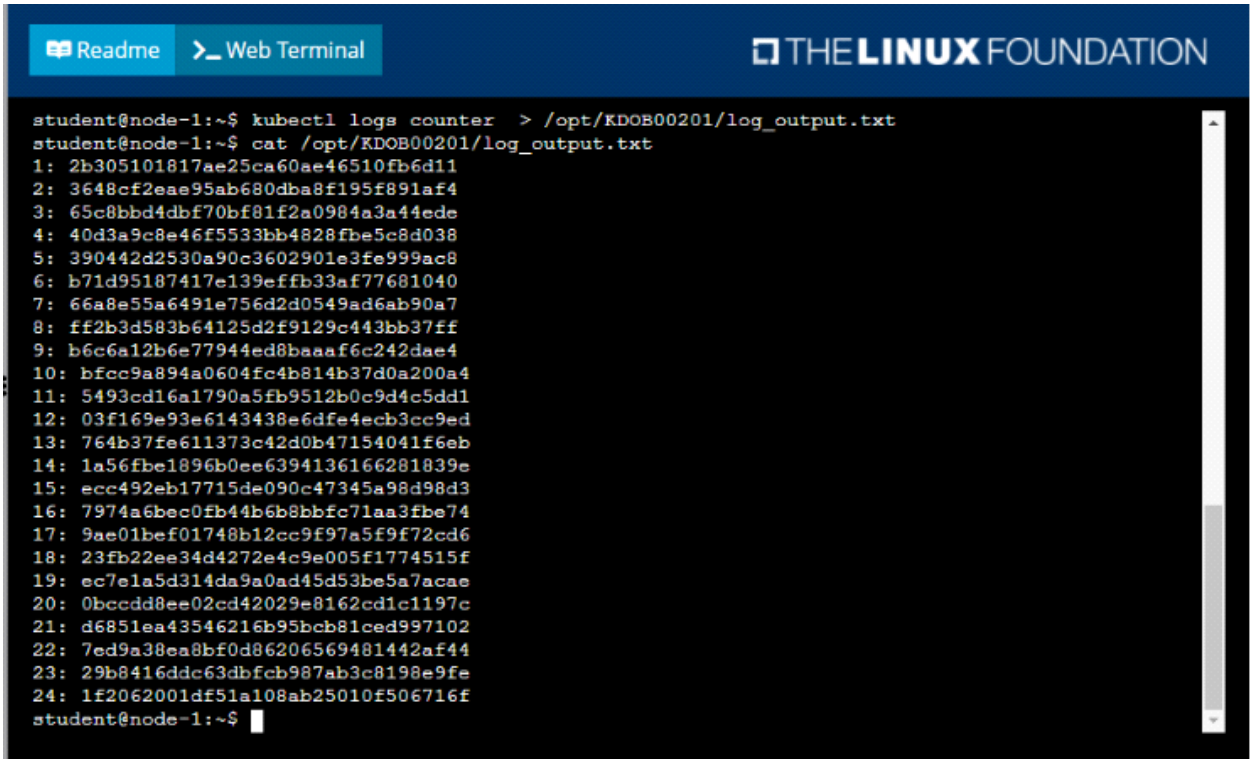
```
student@node-1:~$ kubectl create -f /opt/KDOB00201/counter.yaml
pod/counter created
student@node-1:~$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
counter           1/1     Running   0          10s
liveness-http     1/1     Running   0          6h45m
nginx-101         1/1     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: 40d3a9c8e46f5533bb4828fbe5c8d038
5: 390442d2530a90c3602901e3fe999ac8
6: b71d95187417e139effb33af77681040
7: 66a8e55a6491e756d2d0549ad6ab90a7
8: ff2b3d583b64125d2f9129c443bb37ff
9: b6c6a12b6e77944ed8baaaf6c242dae4
10: bfcc9a894a0604fc4b814b37d0a200a4
student@node-1:~$ kubectl logs counter  > /opt/KDOB00201/log_output.txt
student@node-1:~$
```


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: 40d3a9c8e46f5533bb4828fbe5c8d038
5: 390442d2530a90c3602901e3fe999ac8
6: b71d95187417e139effb33af77681040
7: 66a8e55a6491e756d2d0549ad6ab90a7
8: ff2b3d583b64125d2f9129c443bb37ff
9: b6c6a12b6e77944ed8baaaf6c242dae4
10: bfcc9a894a0604fc4b814b37d0a200a4
11: 5493cd16a1790a5fb9512b0c9d4c5dd1
12: 03f169e93e6143438e6dfe4ecb3cc9ed
13: 764b37fe611373c42d0b47154041f6eb
14: 1a56fbe1896b0ee6394136166281839e
15: ecc492eb17715de090c47345a98d98d3
16: 7974a6bec0fb44b6b8bbfc71aa3fbe74
17: 9ae01bef01748b12cc9f97a5f9f72cd6
18: 23fb22ee34d4272e4c9e005f1774515f
19: ec7e1a5d314da9a0ad45d53be5a7acae
20: 0bccdd8ee02cd42029e8162cd1c1197c
21: d6851ea43546216b95bcb81ced997102
22: 7ed9a38ea8bf0d86206569481442af44
23: 29b8416ddc63dbfcb987ab3c8198e9fe
24: 1f2062001df51a108ab25010f506716f
student@node-1:~$


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: 40d3a9c8e46f5533bb4828fbe5c8d038
5: 390442d2530a90c3602901e3fe999ac8
6: b71d95187417e139effb33af77681040
7: 66a8e55a6491e756d2d0549ad6ab90a7
8: ff2b3d583b64125d2f9129c443bb37ff
9: b6c6a12b6e77944ed8baaaf6c242dae4
10: bfcc9a894a0604fc4b814b37d0a200a4
11: 5493cd16a1790a5fb9512b0c9d4c5dd1
12: 03f169e93e6143438e6dfe4ecb3cc9ed
13: 764b37fe611373c42d0b47154041f6eb
14: 1a56fbe1896b0ee6394136166281839e
15: ecc492eb17715de090c47345a98d98d3
16: 7974a6bec0fb44b6b8bbfc71aa3fbe74
17: 9ae01bef01748b12cc9f97a5f9f72cd6
18: 23fb22ee34d4272e4c9e005f1774515f
19: ec7e1a5d314da9a0ad45d53be5a7acae
20: 0bccdd8ee02cd42029e8162cd1c1197c
21: d6851ea43546216b95bcb81ced997102
22: 7ed9a38ea8bf0d86206569481442af44
23: 29b8416ddc63dbfcb987ab3c8198e9fe
24: 1f2062001df51a108ab25010f506716f
student@node-1:~$

B Solution:

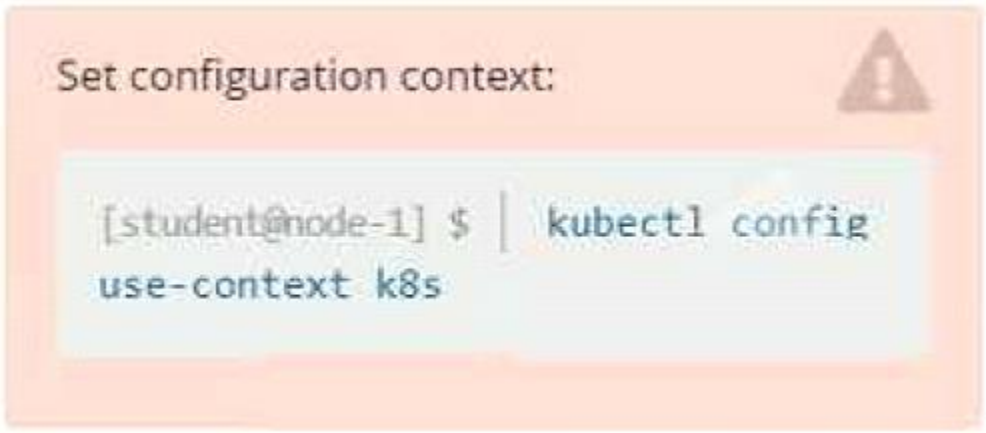
```
student@node-1:~$ kubectl create -f /opt/KDOB00201/counter.yaml
pod/counter created
student@node-1:~$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
counter           1/1     Running   0          10s
liveness-http     1/1     Running   0          6h45m
nginx-101         1/1     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: 40d3a9c8e46f5533bb4828fbe5c8d038
5: 390442d2530a90c3602901e3fe999ac8
6: b71d95187417e139effb33af77681040
7: 66a8e55a6491e756d2d0549ad6ab90a7
8: ff2b3d583b64125d2f9129c443bb37ff
9: b6c6a12b6e77944ed8baaaf6c242dae4
10: bfcc9a894a0604fc4b814b37d0a200a4
student@node-1:~$ kubectl logs counter  > /opt/KDOB00201/log_output.txt
student@node-1:~$
```



Show Answer

Answer : **A**

Q8



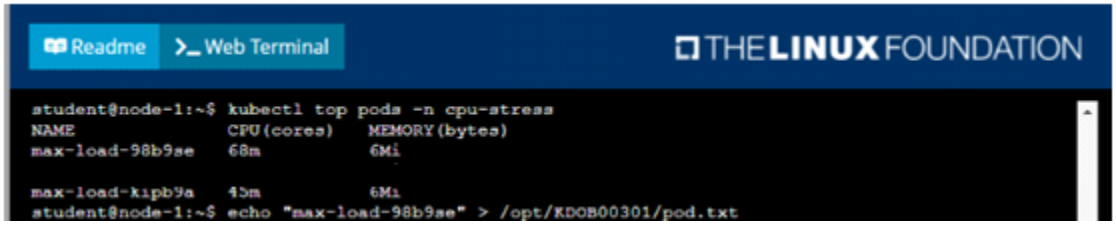
Context

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

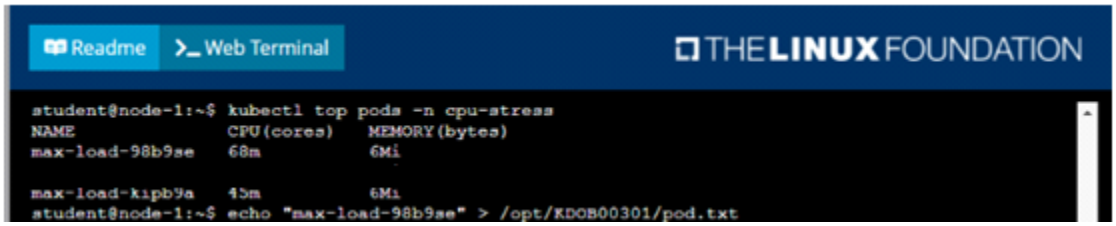
Task

* From the pods running in namespace cpu-stress , write the name only of the pod that is consuming the most CPU to file /opt/KDOBG030l/pod.txt, which has already been created.

A Solution:



B Solution:



Show Answer

Answer : **A**

Q9

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/podl.yml to create a pod named app1 that runs a container named app1cont using image lfccncf/arg-output

with these command line arguments: -lines 56 -F

* Create the pod with the kubect1 command using the YAML file created in the previous step

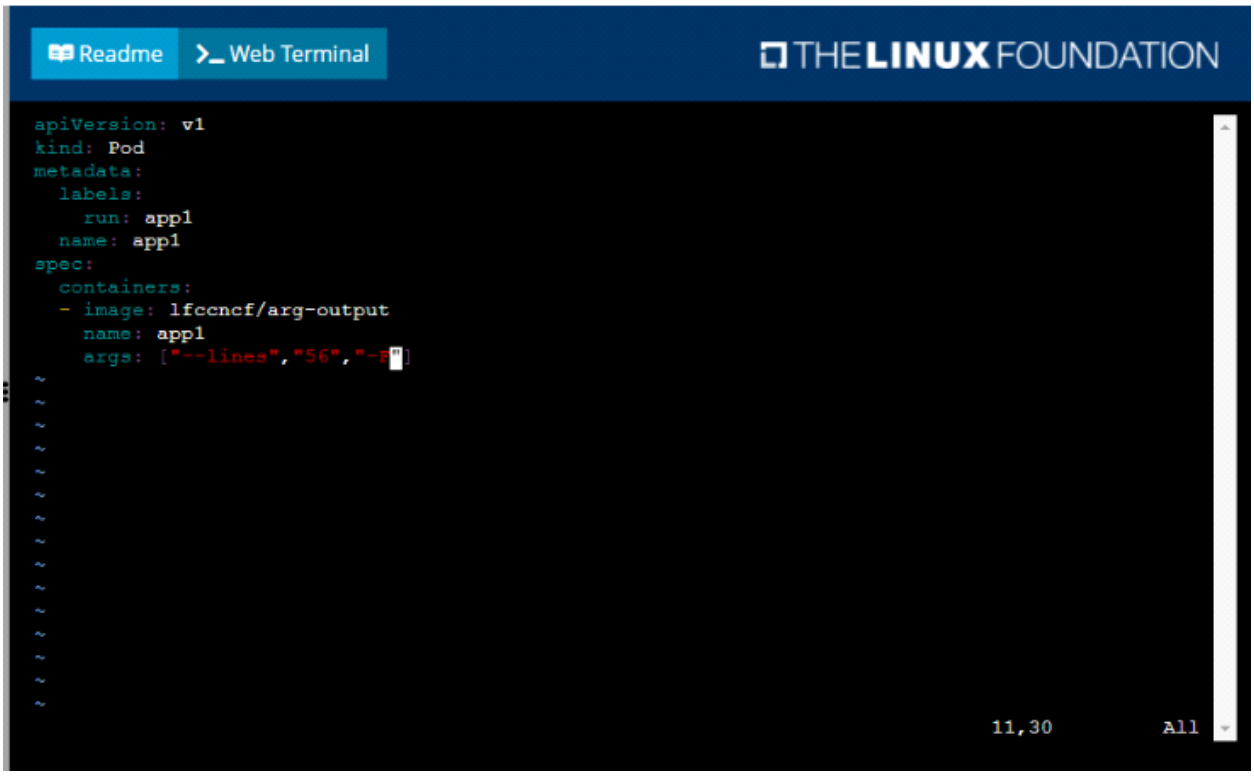
* When the pod is running display summary data about the pod in JSON format using the kubect1 command and redirect the output to a file named /opt/KDPD00101/out1.json

* All of the files you need to work with have been created, empty, for your convenience

A Solution:

```
student@node-1:~$ kubectl run app1 --image=lfccncf/arg-output --dry-run=client -o yaml > /opt/KD
PD00101/pod1.yml
student@node-1:~$ vim /opt/KDPD00101/pod1.yml
```


Readme
Web Terminal
THE LINUX FOUNDATION
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: {}
"/opt/KDPD00101/pod1.yml" 15L, 242C
3,1
All


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","-F"]
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          4m44
liveness-http     1/1     Running             0          6h50
nginx-101         1/1     Running             0          6h51
nginx-configmap   1/1     Running             0          6m21
nginx-secret      1/1     Running             0          11m
poller            1/1     Running             0          6h51
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

THE LINUX FOUNDATION

```
poller            1/1     Running             0          6h51m
student@node-1:~$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
app1              1/1     Running   0          26s
counter           1/1     Running   0          5m5s
liveness-http     1/1     Running   0          6h50m
nginx-101         1/1     Running   0          6h51m
nginx-configmap   1/1     Running   0          6m42s
nginx-secret      1/1     Running   0          12m
poller            1/1     Running   0          6h51m
student@node-1:~$ kubectl delete pod app1
pod "app1" deleted
student@node-1:~$ vim /opt/KDPD00101/pod1.yml
student@node-1:~$ kubectl create -f /opt/KDPD00101/pod1.yml
pod/app1 created
student@node-1:~$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
app1              1/1     Running   0          20s
counter           1/1     Running   0          6m57s
liveness-http     1/1     Running   0          6h52m
nginx-101         1/1     Running   0          6h53m
nginx-configmap   1/1     Running   0          8m34s
nginx-secret      1/1     Running   0          14m
poller            1/1     Running   0          6h53m
student@node-1:~$ kubectl get pod app1 -o json > /opt/KDPD00101/out1.json
student@node-1:~$
student@node-1:~$
```

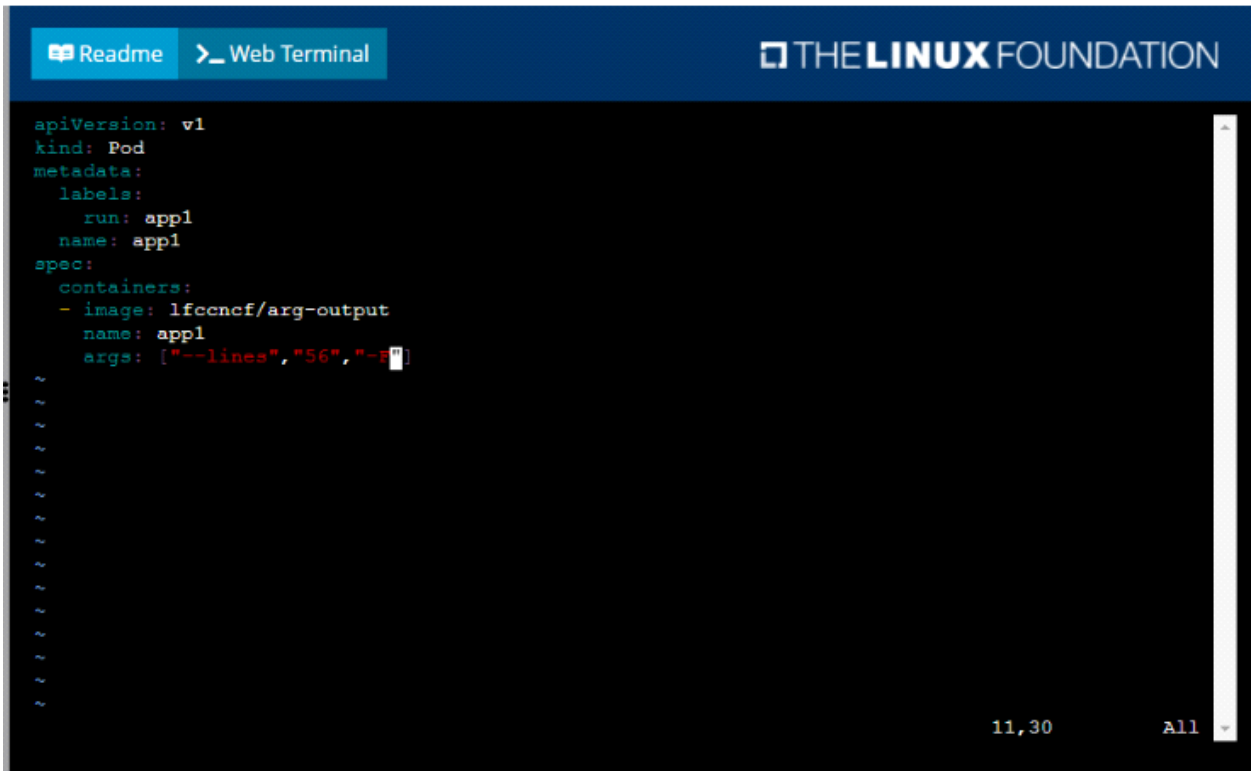
Readme | Web Terminal — THE LINUX FOUNDATION

```
poller            1/1     Running             0          6h51m
student@node-1:~$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
app1              1/1     Running   0          26s
counter           1/1     Running   0          5m5s
liveness-http     1/1     Running   0          6h50m
nginx-101         1/1     Running   0          6h51m
nginx-configmap   1/1     Running   0          6m42s
nginx-secret      1/1     Running   0          12m
poller            1/1     Running   0          6h51m
student@node-1:~$ kubectl delete pod app1
pod "app1" deleted
student@node-1:~$ vim /opt/KDPD00101/pod1.yml
student@node-1:~$ kubectl create -f /opt/KDPD00101/pod1.yml
pod/app1 created
student@node-1:~$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
app1              1/1     Running   0          20s
counter           1/1     Running   0          6m57s
liveness-http     1/1     Running   0          6h52m
nginx-101         1/1     Running   0          6h53m
nginx-configmap   1/1     Running   0          8m34s
nginx-secret      1/1     Running   0          14m
poller            1/1     Running   0          6h53m
student@node-1:~$ kubectl get pod app1 -o json > /opt/KDPD00101/out1.json
student@node-1:~$
student@node-1:~$
```

B Solution:

```
student@node-1:~$ kubectl run app1 --image=lfccncf/arg-output --dry-run=client -o yaml > /opt/KD
PD00101/pod1.yml
student@node-1:~$ vim /opt/KDPD00101/pod1.yml
```


Readme
Web Terminal
THE LINUX FOUNDATION
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: {}
"/opt/KDPD00101/pod1.yml" 15L, 242C
3,1
All


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","-F"]
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 — THE LINUX FOUNDATION

```
poller              1/1     Running             0          6h51m
student@node-1:~$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
app1                1/1     Running   0          26s
counter             1/1     Running   0          5m5s
liveness-http       1/1     Running   0          6h50m
nginx-101           1/1     Running   0          6h51m
nginx-configmap     1/1     Running   0          6m42s
nginx-secret        1/1     Running   0          12m
poller              1/1     Running   0          6h51m
student@node-1:~$ kubectl delete pod app1
pod "app1" deleted
student@node-1:~$ vim /opt/KDPD00101/pod1.yml
student@node-1:~$ kubectl create -f /opt/KDPD00101/pod1.yml
pod/app1 created
student@node-1:~$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
app1                1/1     Running   0          20s
counter             1/1     Running   0          6m57s
liveness-http       1/1     Running   0          6h52m
nginx-101           1/1     Running   0          6h53m
nginx-configmap     1/1     Running   0          8m34s
nginx-secret        1/1     Running   0          14m
poller              1/1     Running   0          6h53m
student@node-1:~$ kubectl get pod app1 -o json > /opt/KDPD00101/out1.json
student@node-1:~$
student@node-1:~$
```

Show Answer

Answer : **A**

Q10

Task

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

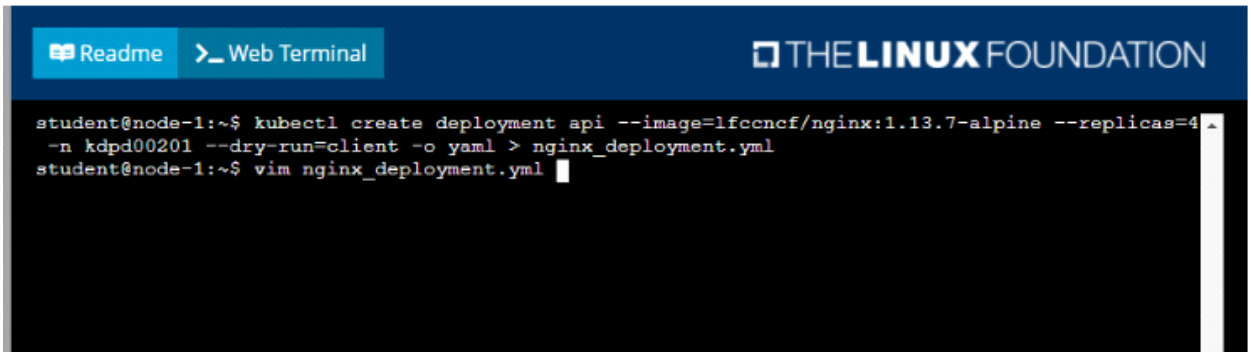
* Run the deployment in the kdpd00201 namespace. The namespace has already been created

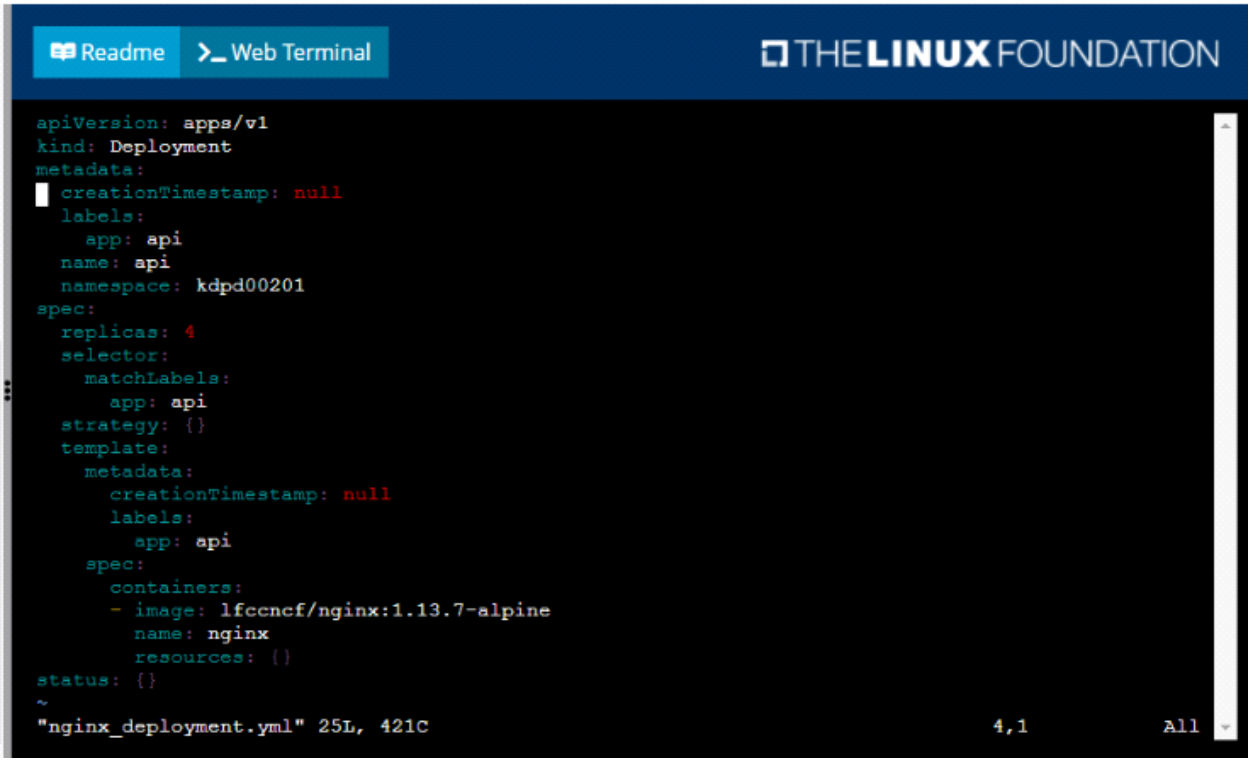
* Name the deployment frontend and configure with 4 replicas

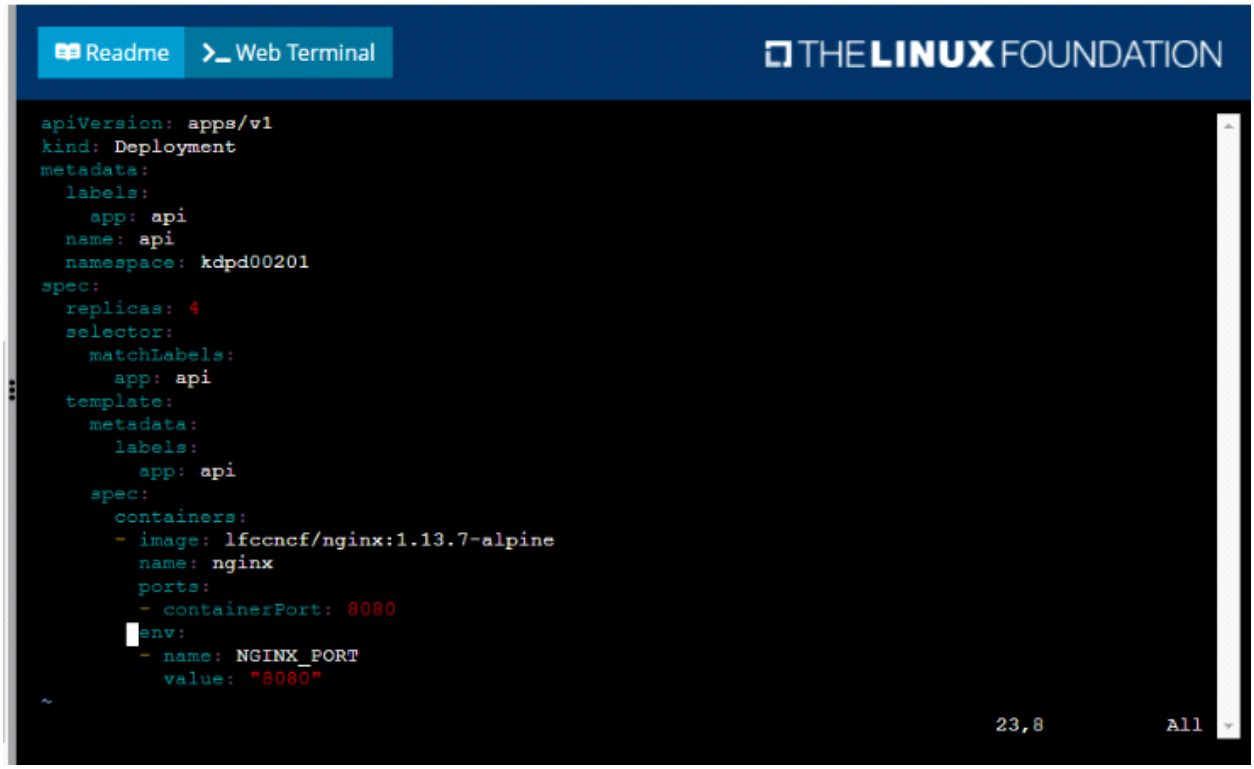
* Configure the pod with a container image of lfccncf/nginx:1.13.7

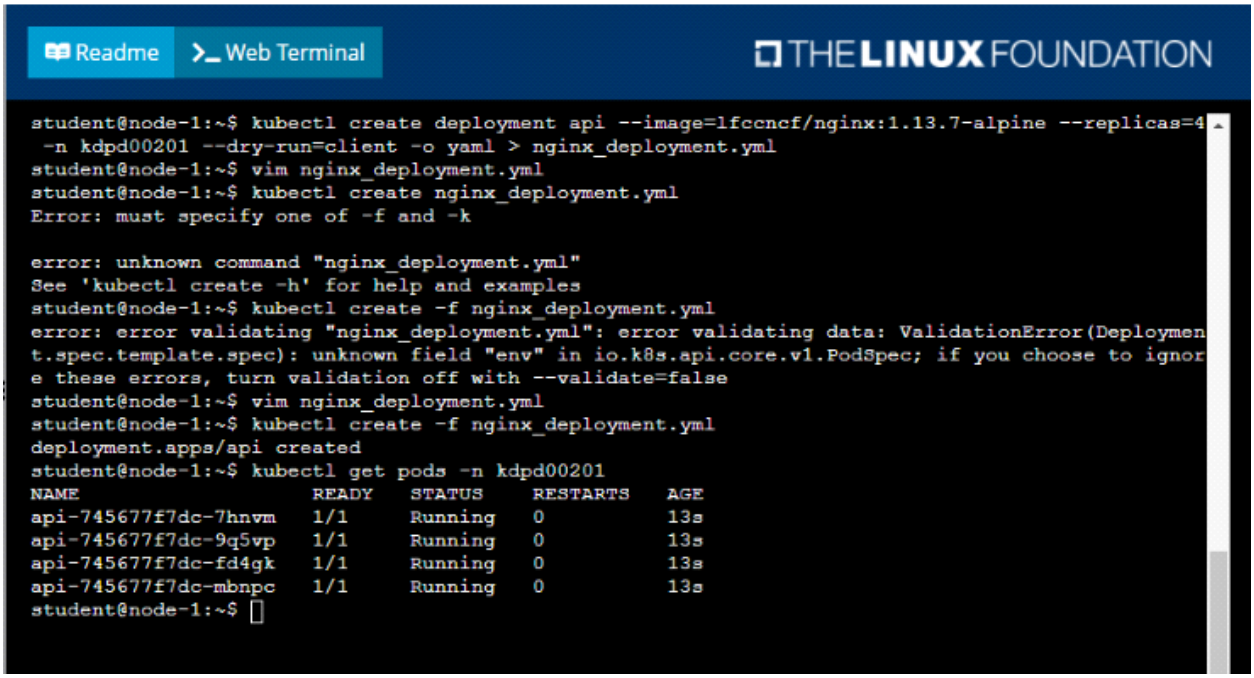
* Set an environment variable of NGINX__PORT=8080 and also expose that port for the container above

A Solution:

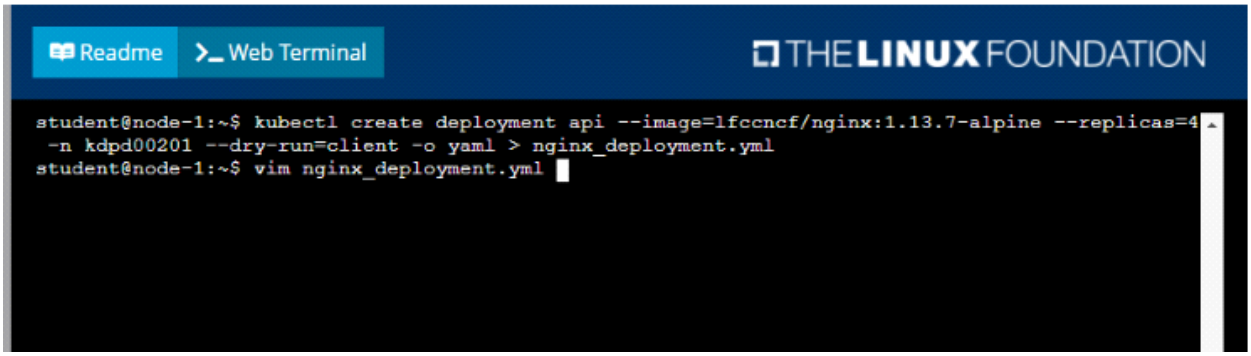


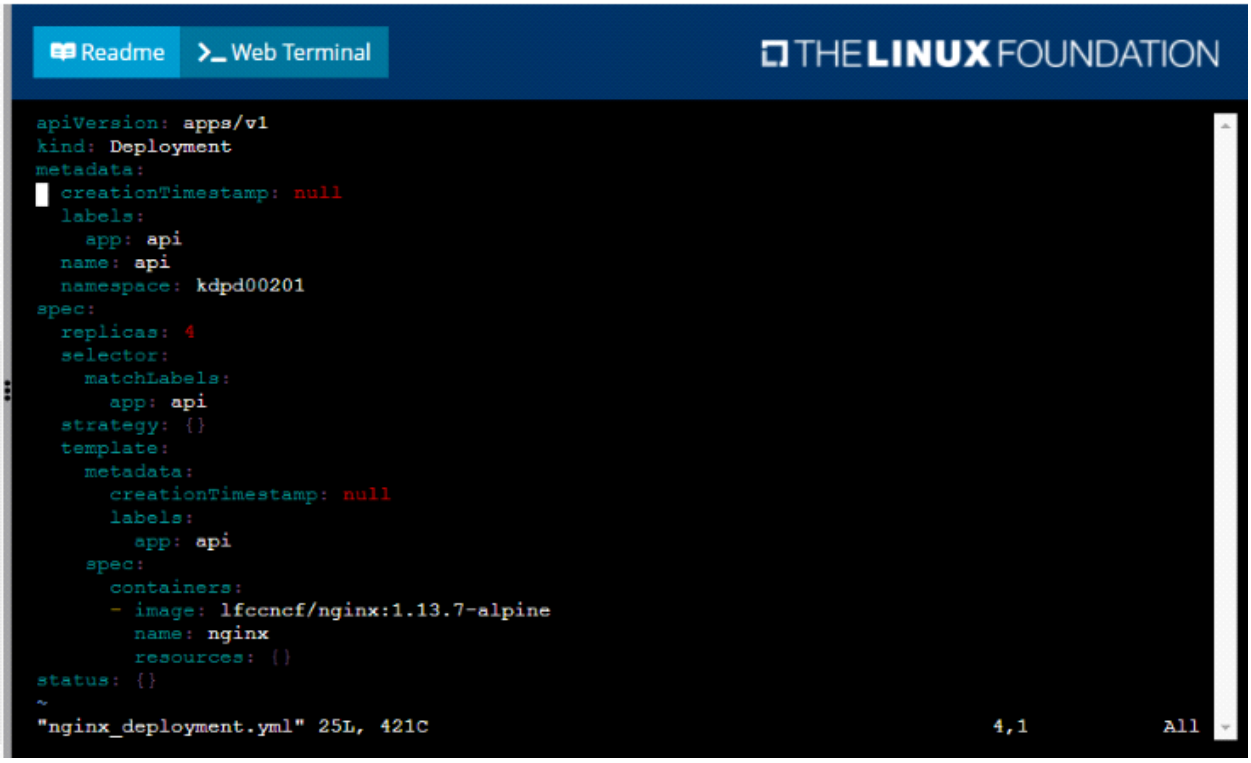

Readme
Web Terminal
THE LINUX FOUNDATION
apiVersion: apps/v1
kind: Deployment
metadata:
creationTimestamp: null
labels:
app: api
name: api
namespace: kdpd00201
spec:
replicas: 4
selector:
matchLabels:
app: api
strategy: {}
template:
metadata:
creationTimestamp: null
labels:
app: api
spec:
containers: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

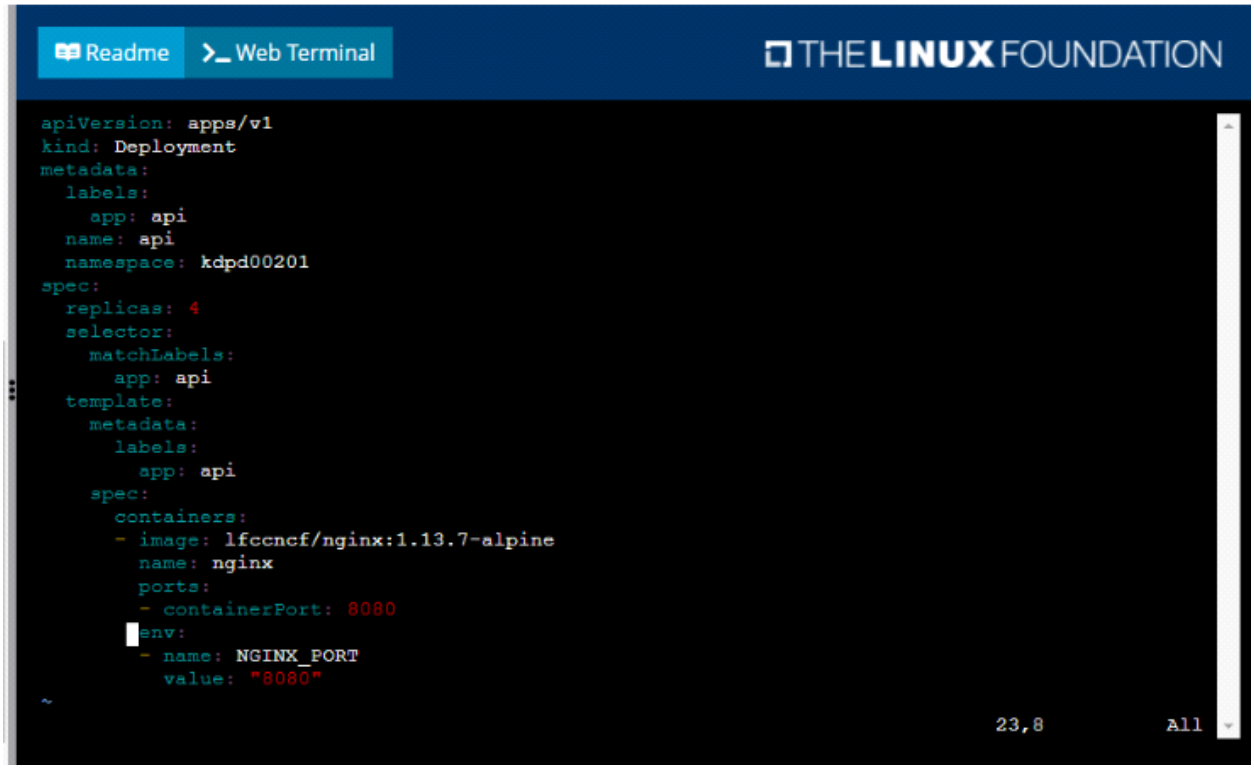

Readme
Web Terminal
THE LINUX FOUNDATION
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: api
name: api
namespace: kdpd00201
spec:
replicas: 4
selector:
matchLabels:
app: api
template:
metadata:
labels:
app: api
spec:
containers:
- image: lfccncf/nginx:1.13.7-alpine
name: nginx
ports:
- containerPort: 8080
env:
- name: NGINX_PORT
value: "8080"
~
23,8
All

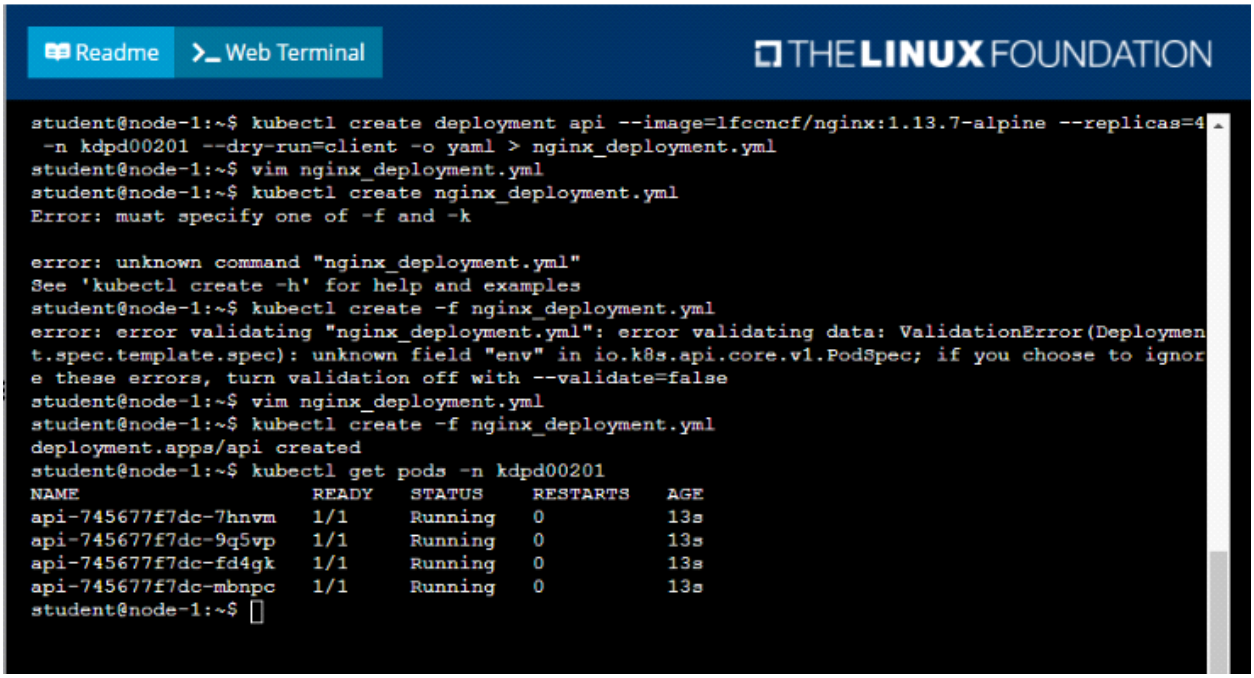

Readme
Web Terminal
THE LINUX FOUNDATION
student@node-1:~$ kubectl create deployment api --image=lfccncf/nginx:1.13.7-alpine --replicas=4
-n kdpd00201 --dry-run=client -o yaml > nginx_deployment.yml
student@node-1:~$ vim nginx_deployment.yml
student@node-1:~$ kubectl create nginx_deployment.yml
Error: 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(Deploymen
t.spec.template.spec): unknown field "env" in io.k8s.api.core.v1.PodSpec; if you choose to ignor
e 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-7hnvm 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:~$

B Solution:


Readme
Web Terminal
THE LINUX FOUNDATION
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



Show Answer

Answer : **B**

Q11

Exhibit:

Context

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

Task

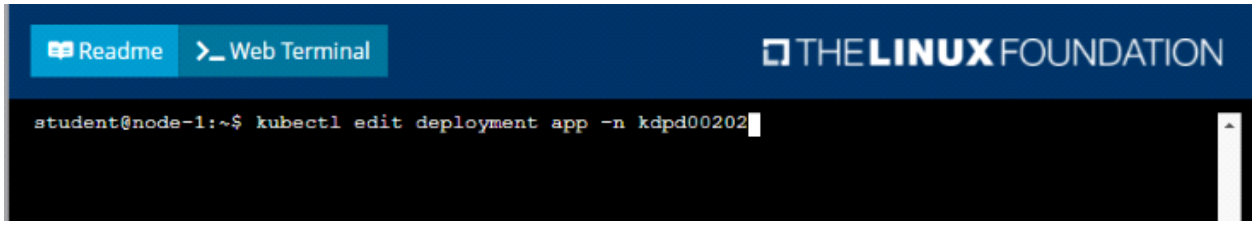
Please complete the following:

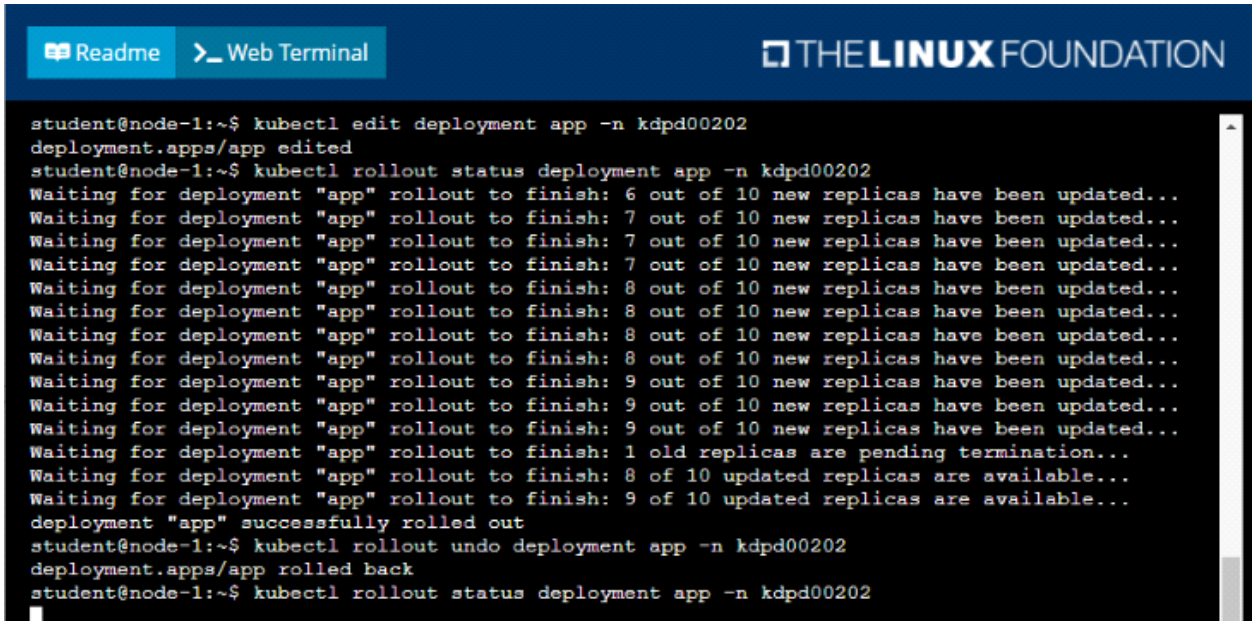
* Update the app deployment in the kdpd00202 namespace with a maxSurge of 5% and a maxUnavailable of 2%

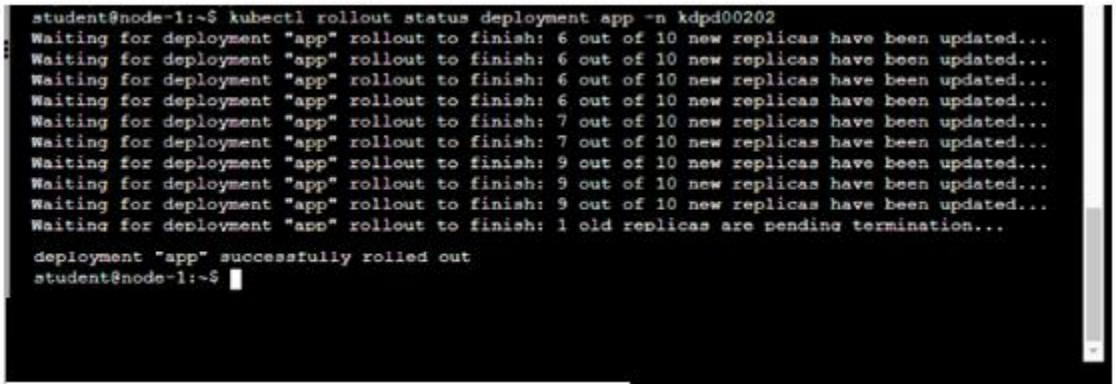
* Perform a rolling update of the web1 deployment, changing the lfccncf/ngmx image version to 1.13

* Roll back the app deployment to the previous version

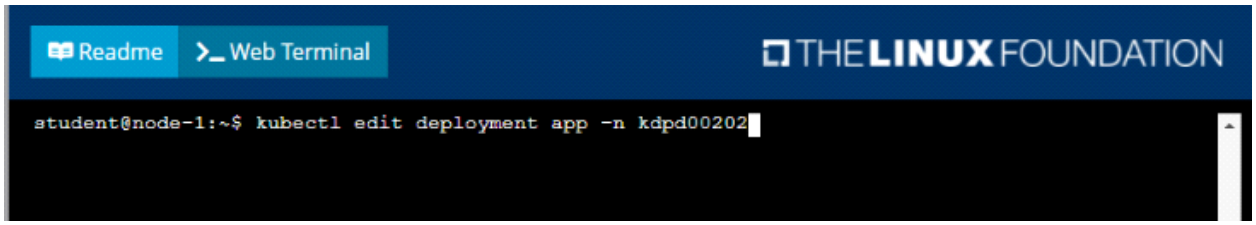
Solution:



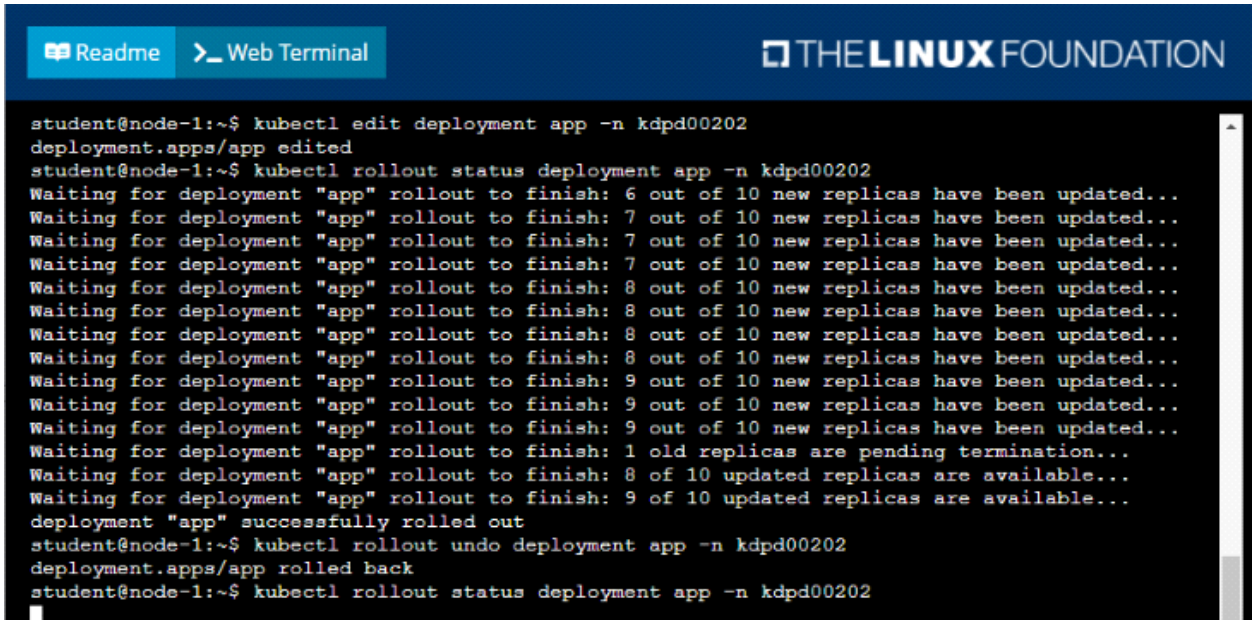

Readme
Web Terminal
THE LINUX FOUNDATION
student@node-1:~$ kubectl edit deployment app -n kdpd00202
deployment.apps/app edited
student@node-1:~$ kubectl rollout status deployment app -n kdpd00202
Waiting for deployment "app" rollout to finish: 6 out of 10 new replicas have been updated...
Waiting for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

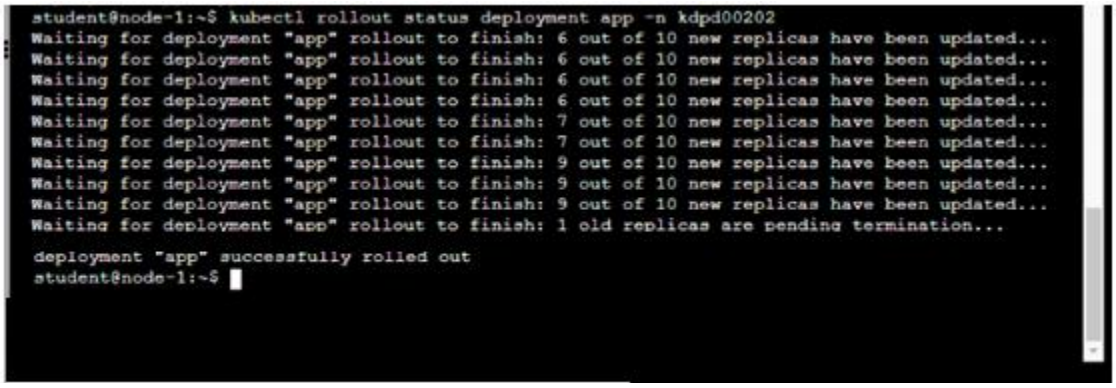

student@node-1:~$ kubectl rollout status deployment app -n kdpd00202
Waiting for deployment "app" rollout to finish: 6 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: 6 out of 10 new replicas have been updated...
Waiting for deployment "app" rollout to finish: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

B Solution:


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


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

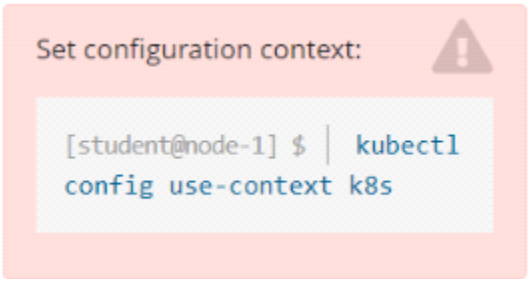

Readme
Web Terminal
THE LINUX FOUNDATION
student@node-1:~$ kubectl edit deployment app -n kdpd00202
deployment.apps/app edited
student@node-1:~$ kubectl rollout status deployment app -n kdpd00202
Waiting for deployment "app" rollout to finish: 6 out of 10 new replicas have been updated...
Waiting for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



Hide Answer

Suggested Answer: *A*

Q12



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**

lfccncf/busybox:1 container, named logger-dev

*** includes a sidecar lfccncf/fluentd:v0.12 container, named adapter-zen**

*** Mounts a shared volume /tmp/log on both containers, which does not persist when the pod is deleted**

*** Instructs the logger-dev**

container to run the command

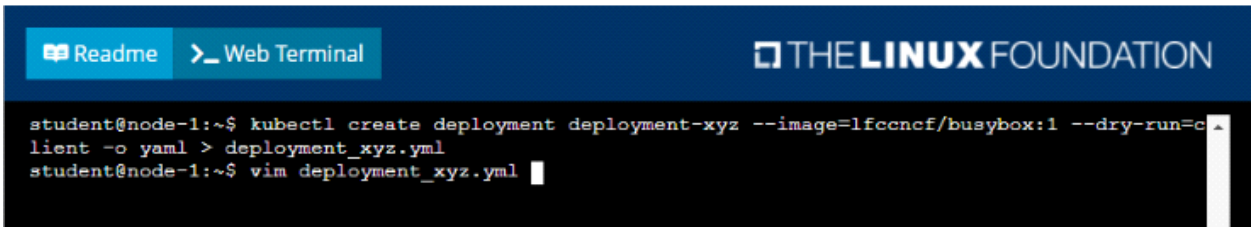
```
while true; do
echo "i luv cncf" >> /
tmp/log/input.log;
sleep 10;
done
```

which should output logs to /tmp/log/input.log in plain text format, with example values:

```
i luv cncf
i luv cncf
i luv cncf
```

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

ASolution:




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: lfccncf/busybox:1
name: busybox
resources: {}
status: {}
~
~
"deployment_xyz.yml" 24L, 434C
3,1
All


Readme
Web Terminal
THE LINUX FOUNDATION
kind: Deployment
metadata:
labels:
app: deployment-xyz
name: deployment-xyz
spec:
replicas: 1
selector:
matchLabels:
app: deployment-xyz
template:
metadata:
labels:
app: deployment-xyz
spec:
volumes:
- name: myvol1
emptyDir: {}
containers:
- image: lfccncf/busybox:1
name: logger-dev
volumeMounts:
- name: myvol1
mountPath: /tmp/log
- image: lfccncf/fluentd:v0.12
name: adapter-zen
3 lines yanked
27,22
Bot

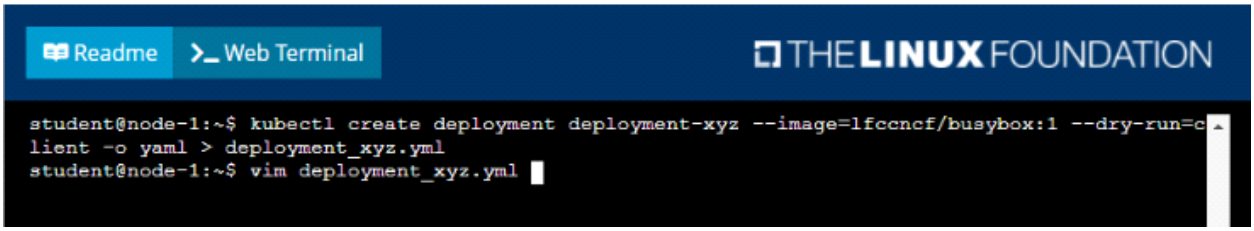
Readme | Web Terminal — THE LINUX FOUNDATION

```
    metadata:
      labels:
        app: deployment-xyz
    spec:
      volumes:
      - name: myvol1
        emptyDir: {}
      - name: myvol2
        configMap:
          name: logconf
      containers:
      - image: lfccncf/busybox:1
        name: logger-dev
        command: ["/bin/sh","-c","while [ true ]; do echo 'i luv cncf' >> /tmp/log/input.log; sl
eep 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/etc
                                                                    37,33         Bot
```

```
student@node-1:~$ kubectl create -f deployment_xyz.yml
deployment.apps/deployment-xyz created
student@node-1:~$ kubectl get deployment
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz   0/1     1            0           5s
student@node-1:~$ kubectl get deployment
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz   0/1     1            0           9s
student@node-1:~$ kubectl get deployment
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz   1/1     1            1           12s
student@node-1:~$
```


```
student@node-1:~$ kubectl create -f deployment_xyz.yml
deployment.apps/deployment-xyz created
student@node-1:~$ kubectl get deployment
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz   0/1     1            0           5s
student@node-1:~$ kubectl get deployment
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz   0/1     1            0           9s
student@node-1:~$ kubectl get deployment
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
deployment-xyz   1/1     1            1           12s
student@node-1:~$
```


BSolution:


Readme
Web Terminal
THE LINUX FOUNDATION
student@node-1:~$ kubectl create deployment deployment-xyz --image=lfccncf/busybox:1 --dry-run=client -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: lfccncf/busybox:1
name: busybox
resources: {}
status: {}
~
~
"deployment_xyz.yml" 24L, 434C
3,1
All


Readme
Web Terminal
THE LINUX FOUNDATION
kind: Deployment
metadata:
labels:
app: deployment-xyz
name: deployment-xyz
spec:
replicas: 1
selector:
matchLabels:
app: deployment-xyz
template:
metadata:
labels:
app: deployment-xyz
spec:
volumes:
- name: myvol1
emptyDir: {}
containers:
- image: lfccncf/busybox:1
name: logger-dev
volumeMounts:
- name: myvol1
mountPath: /tmp/log
- image: lfccncf/fluentd:v0.12
name: adapter-zen
3 lines yanked
27,22
Bot


Readme
Web Terminal
THE LINUX FOUNDATION
metadata:
labels:
app: deployment-xyz
spec:
volumes:
- name: myvol1
emptyDir: {}
- name: myvol2
configMap:
name: logconf
containers:
- image: lfccncf/busybox:1
name: logger-dev
command: ["/bin/sh","-c","while [true]; do echo 'i luv cncf' >> /tmp/log/input.log; sl
eep 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/etc
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:~$
```

Q13

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 ana deploy in namespace kdsn00l01 a service that accomplishes the following:

* Exposes the service on TCP port 8080

* is mapped to me pods defined by the specification of kdsn00l01-deployment

* Is of type NodePort

* Has a name of cherry

A Solution:

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

Readme | Web Terminal — THE LINUX FOUNDATION

```
# 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
```

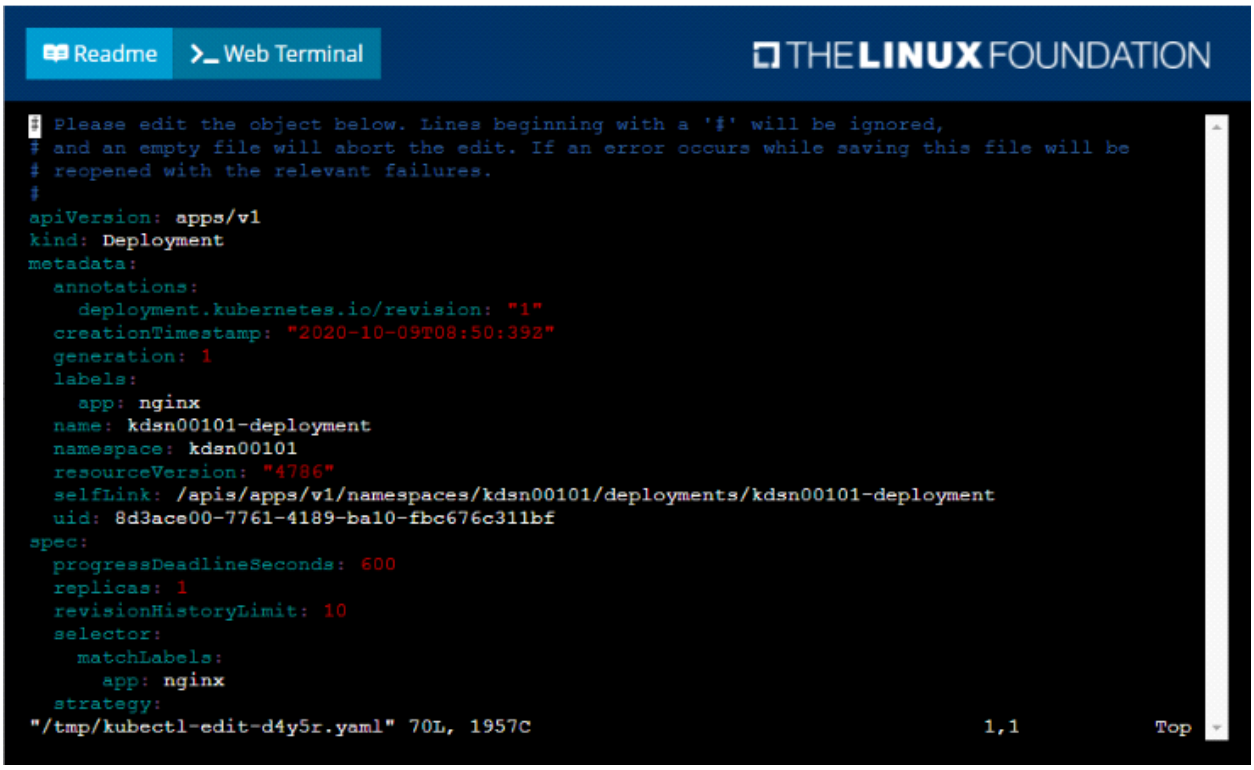
Readme | Web Terminal — THE LINUX FOUNDATION

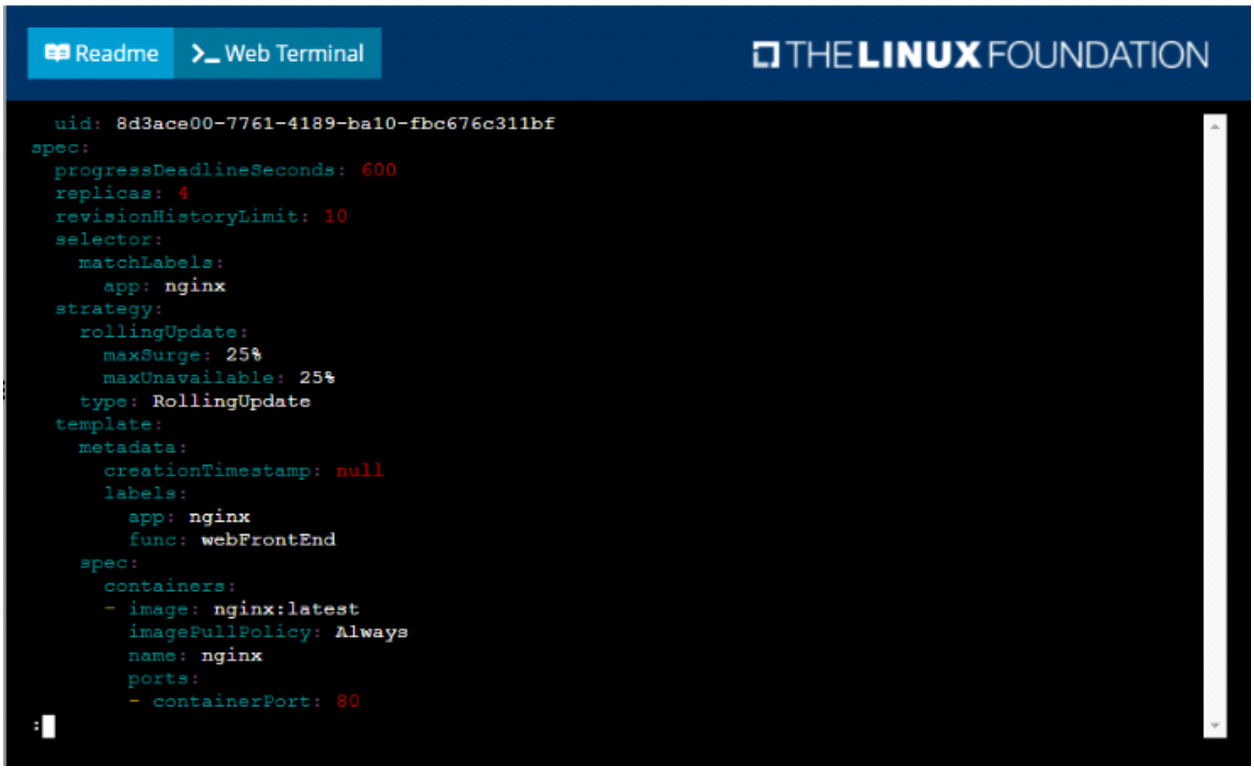
```
  uid: 8d3ace00-7761-4189-ba10-fbc676c311bf
spec:
  progressDeadlineSeconds: 600
  replicas: 4
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      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
```

B Solution:

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





Show Answer

Answer : **A**

Q14

Context

Developers occasionally need to submit pods that run periodically.

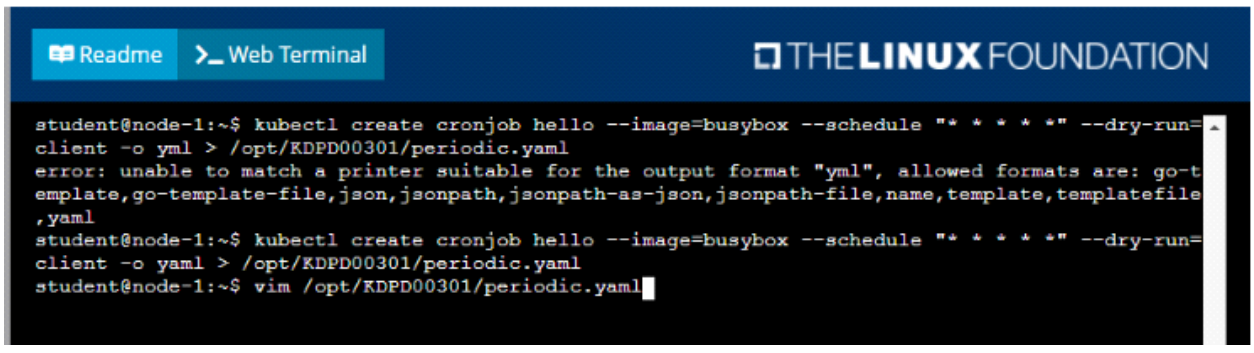
Task

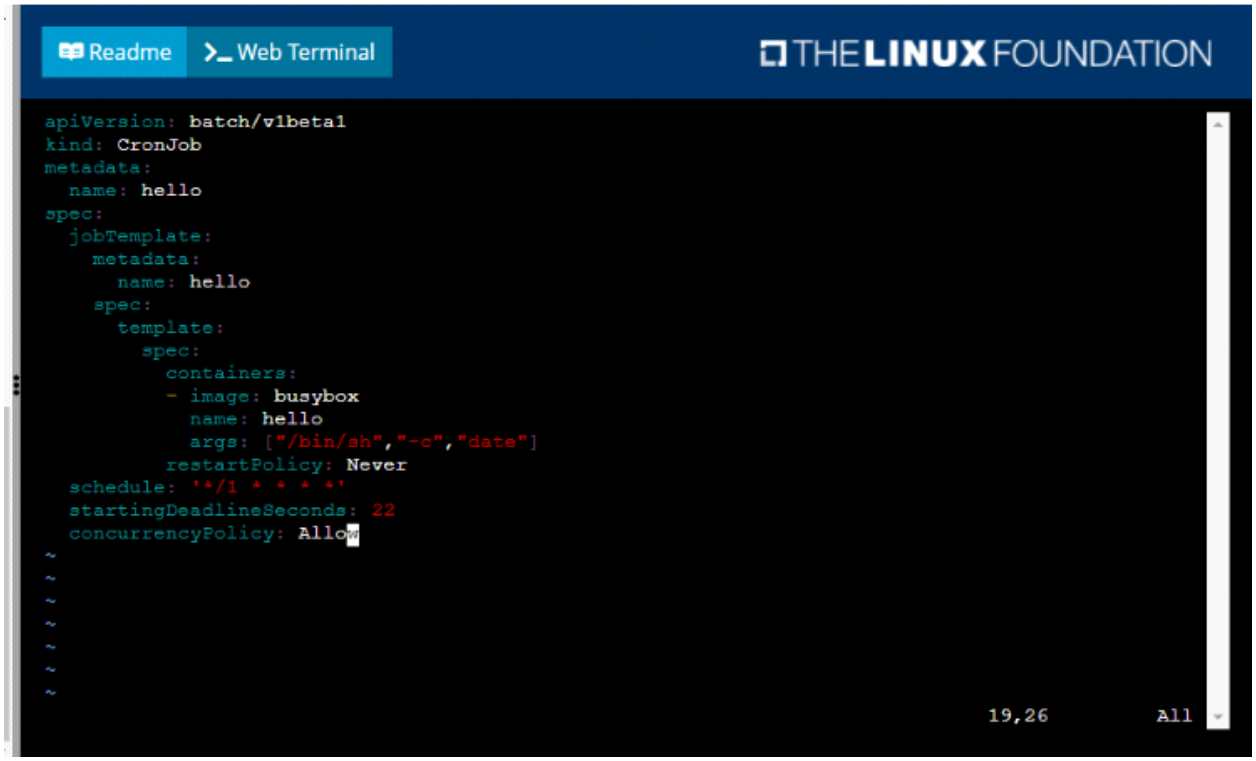
Follow the steps below to create a pod that will start at a predetermined time and]which runs to completion only once each time it is started:

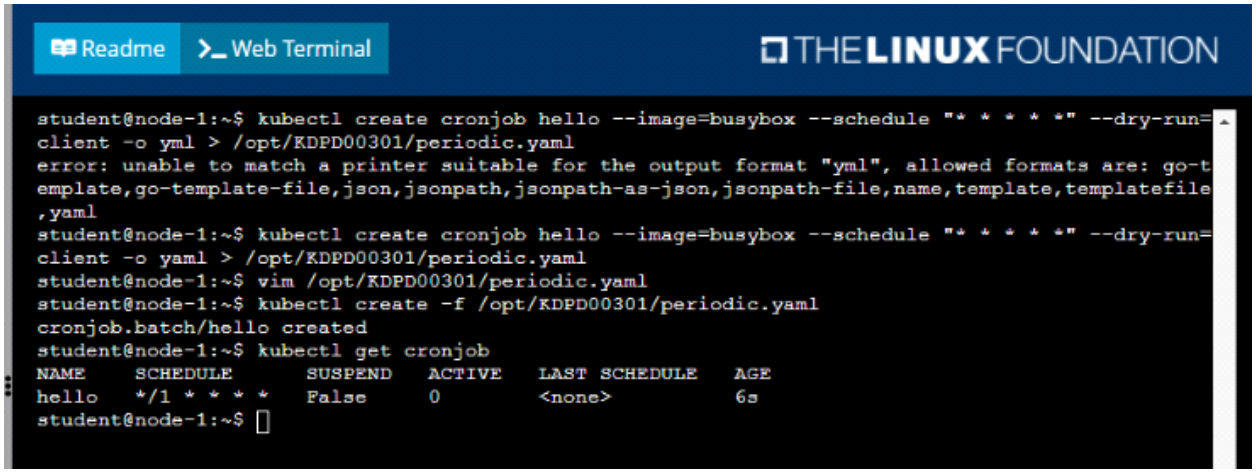
* Create a YAML formatted Kubernetes manifest /opt/KDPD00301/periodic.yaml that runs the following shell command: date in a single busybox container. The command should run every minute and must complete within 22 seconds or be terminated oy Kubernetes. The Cronjob namp and container name should both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

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

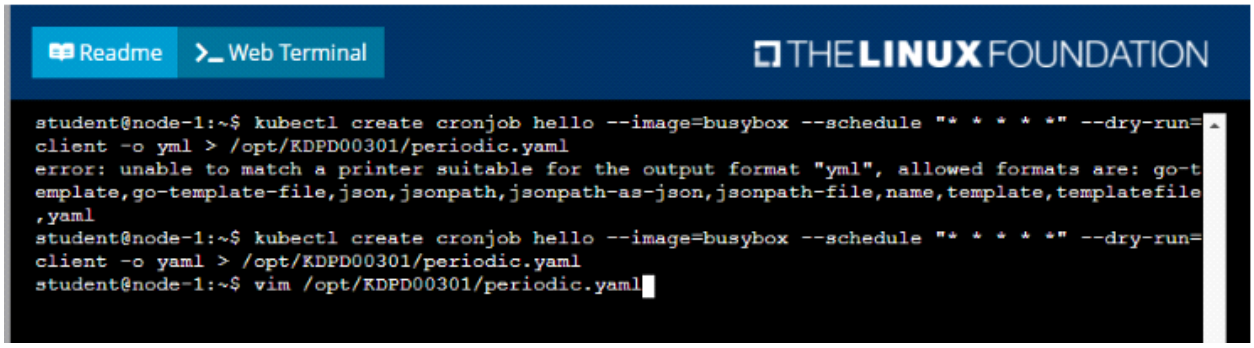
Solution:

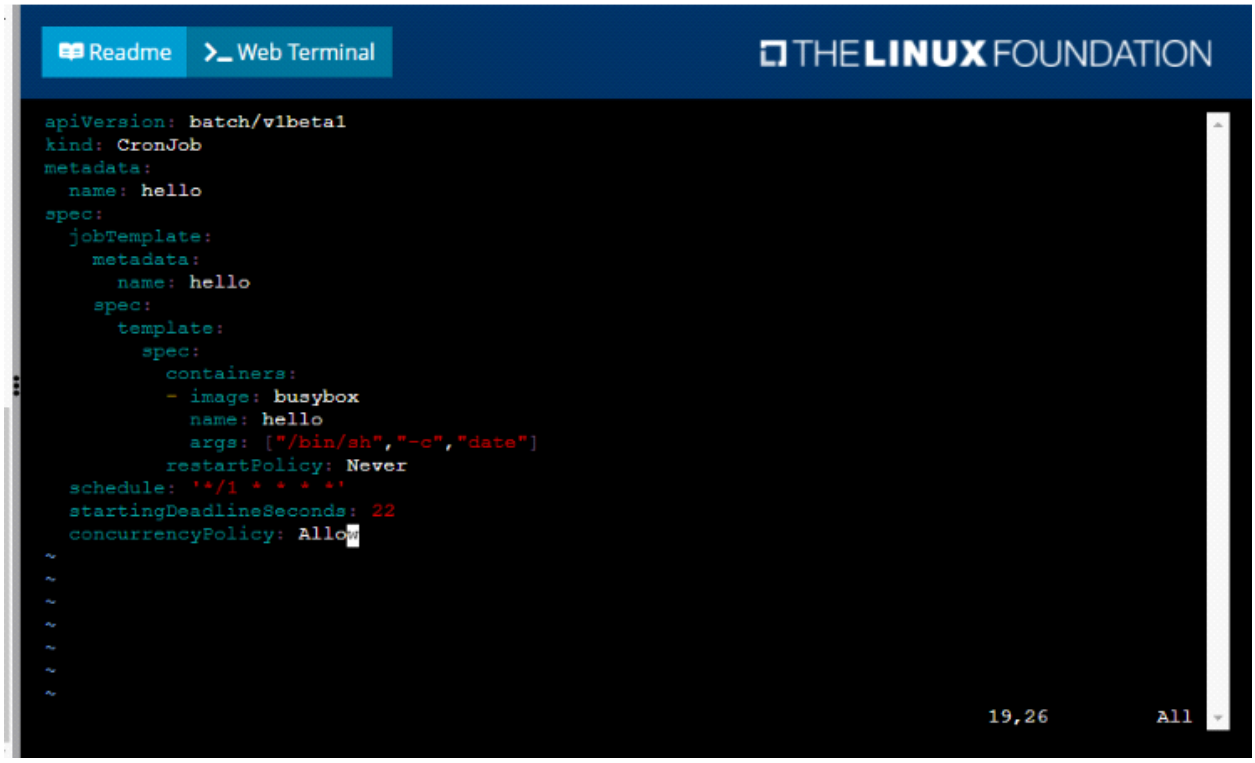



Readme
Web Terminal
THE LINUX FOUNDATION
apiVersion: batch/v1beta1
kind: CronJob
metadata:
name: hello
spec:
jobTemplate:
metadata:
name: hello
spec:
template:
spec:
containers:
- image: busybox
name: hello
args: ["/bin/sh","-c","date"]
restartPolicy: Never
schedule: '*/1 * * * *'
startingDeadlineSeconds: 22
concurrencyPolicy: Allow
19,26
All


Readme
Web Terminal
THE LINUX FOUNDATION
student@node-1:~$ kubectl create cronjob hello --image=busybox --schedule "* * * * *" --dry-run=client -o yml > /opt/KDPD00301/periodic.yaml
error: unable to match a printer suitable for the output format "yml", allowed formats are: go-template,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:~$

Solution:

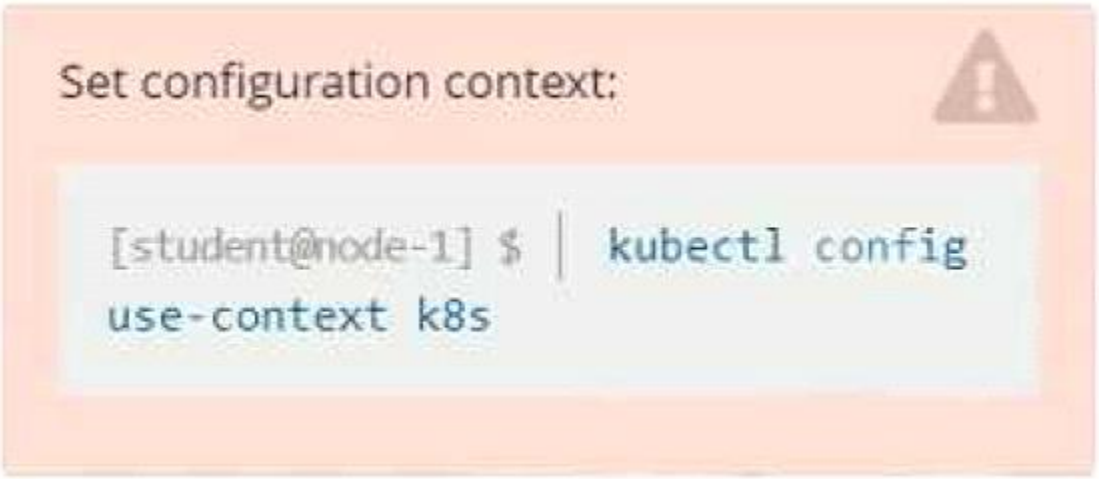




Hide Answer

Suggested Answer: *A*

Q15



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

ASolution:

```
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

Solution:
apiVersion: apps/v1
kind: Deployment
metadata:
name: my-nginx
spec:
selector:
matchLabels:
run: my-nginx
- 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

Hide Answer
Suggested Answer: *A*

Q16

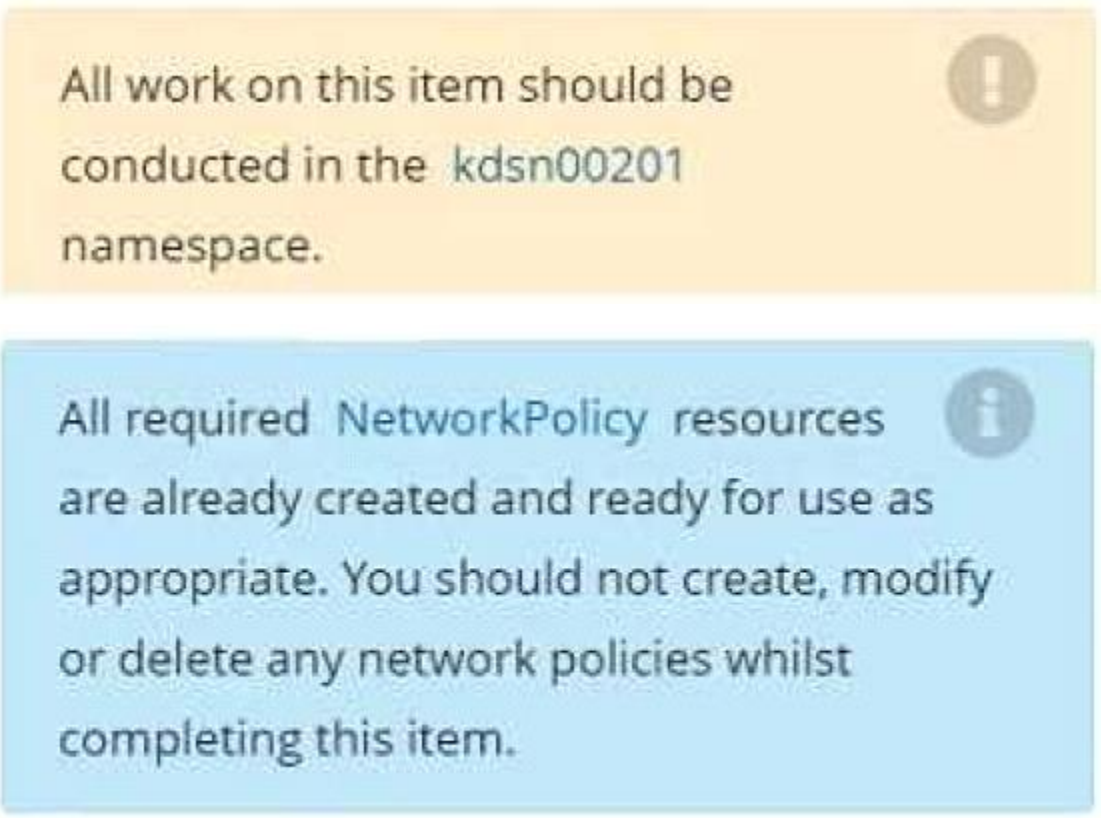
Task

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

Q17

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.



Q18

Context

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

Task

Perform the following tasks:

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

```
<namespace>/<pod>
```

The output file has already been created

*** Store the associated error events to a file /opt/KDOB00401/error.txt, The output file has already been created. You will need to use the -o wide output specifier with your command**

*** Fix the issue.**

ASolution:
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 thatRESTARTShas been incremented:
NAME READY STATUS RESTARTS AGE
liveness-exec 1/1 Running 1 m

BSolution:
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'
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 thatRESTARTShas been incremented:
NAME READY STATUS RESTARTS AGE
liveness-exec 1/1 Running 1 m

Q19

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 PersistenetVolume.

* Create a PefsissentVolumeClaim named task-pv-claim that requests a volume of at least 100Mi and specifies an access mode of ReadWriteOnce

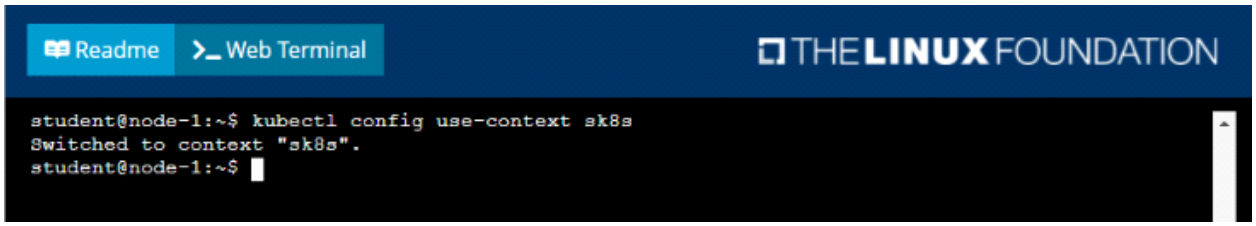
* Create a pod that uses the PersistentVolmeClaim as a volume with a label app: my-storage-app mounting the resulting volume to a mountPath /usr/share/nginx/html inside the pod

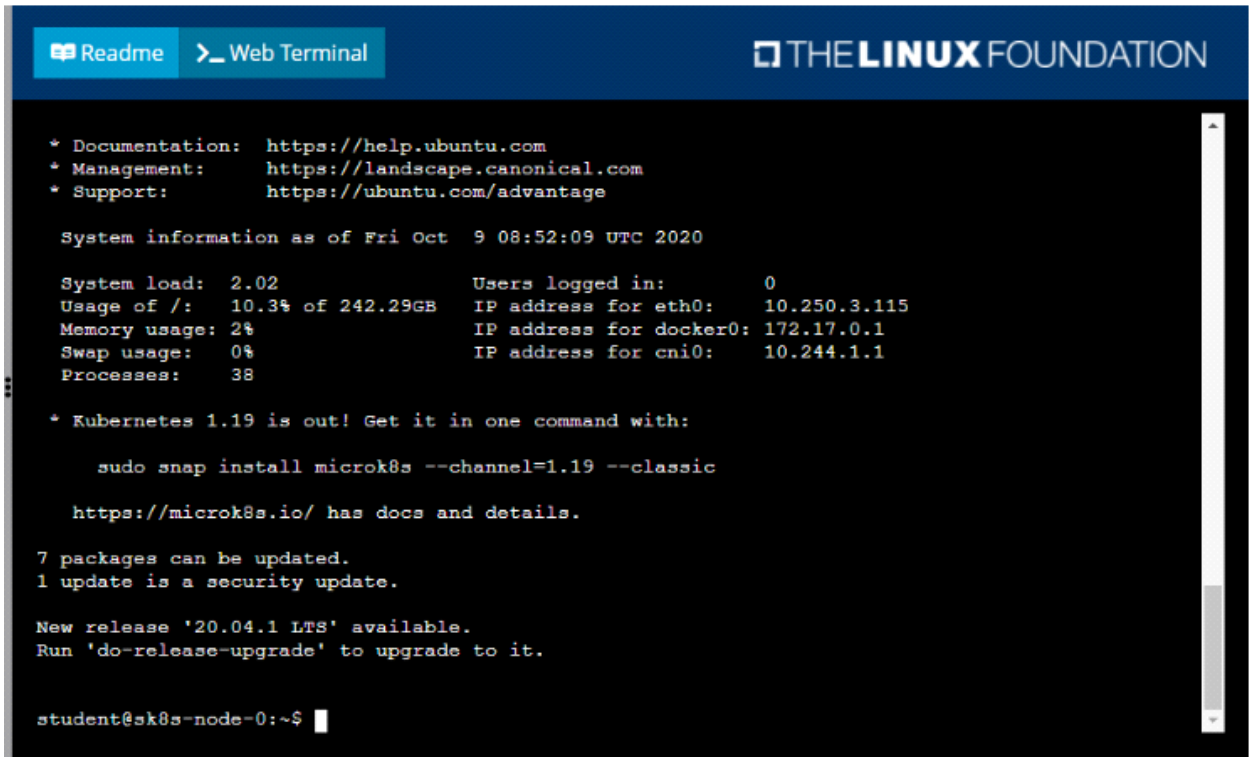
You can access sk8s-node-0 by issuing the following command:

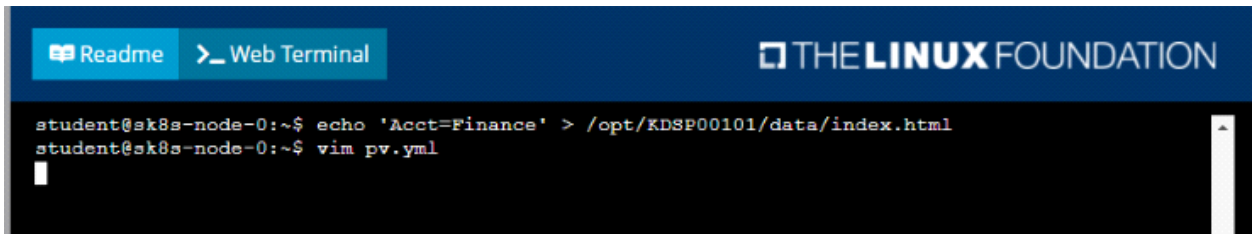
```
[student@node-1] $ ssh sk8s-node-0
```

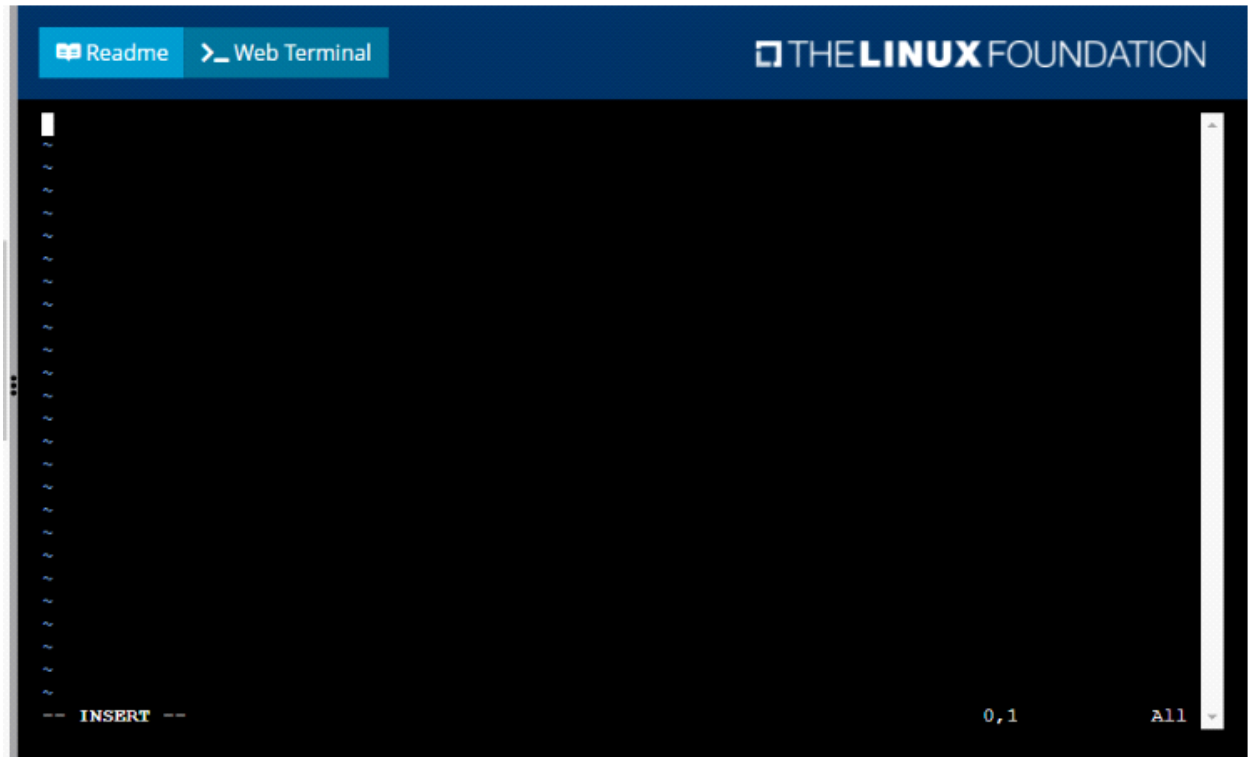
Ensure that you return to the base node (with hostname node-1) once you have completed your work on sk8s-node-0 copy

Solution:




Readme
Web Terminal
THE LINUX FOUNDATION
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information as of Fri Oct 9 08:52:09 UTC 2020
System load: 2.02
Usage of /: 10.3% of 242.29GB
Memory usage: 2%
Swap usage: 0%
Processes: 38
Users logged in: 0
IP address for eth0: 10.250.3.115
IP address for docker0: 172.17.0.1
IP address for cni0: 10.244.1.1
* Kubernetes 1.19 is out! Get it in one command with:
sudo snap install microk8s --channel=1.19 --classic
https://microk8s.io/ has docs and details.
7 packages can be updated.
1 update is a security update.
New release '20.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
student@sk8s-node-0:~$


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




```
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                   STO
RAGECLASS   REASON   AGE
task-pv-volume   1Gi        RWO            Retain           Bound    default/task-pv-claim   sto
rage                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
```

Readme | Web Terminal — THE LINUX FOUNDATION

```
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
~
~
~
~
~
~
~
~
~
                                                    17,32         All
```

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

Readme | Web Terminal — THE LINUX FOUNDATION

```
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:~$
```

Readme | Web Terminal — THE LINUX FOUNDATION

```
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:~$
```

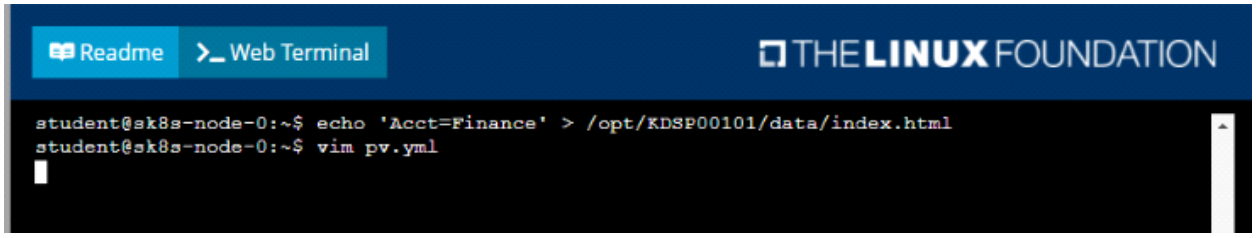
B Solution:

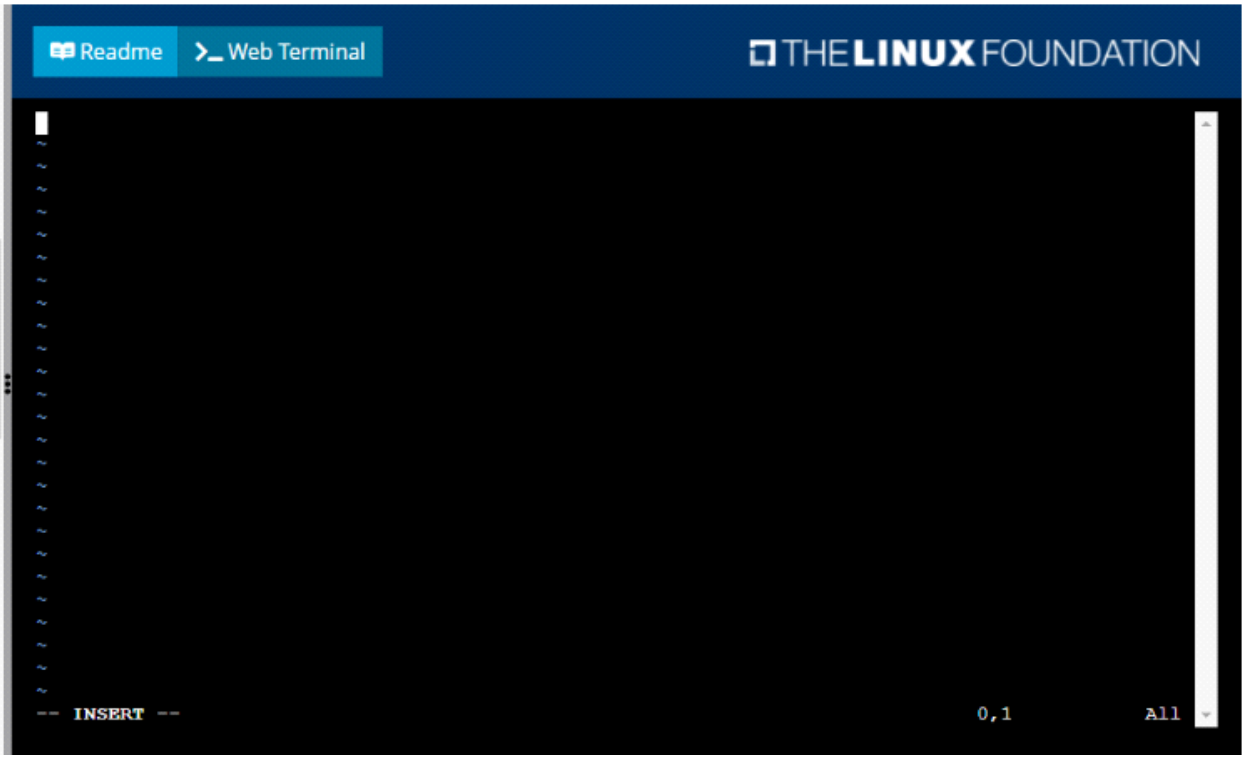
Readme | Web Terminal — THE LINUX FOUNDATION

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

Readme | Web Terminal — THE LINUX FOUNDATION

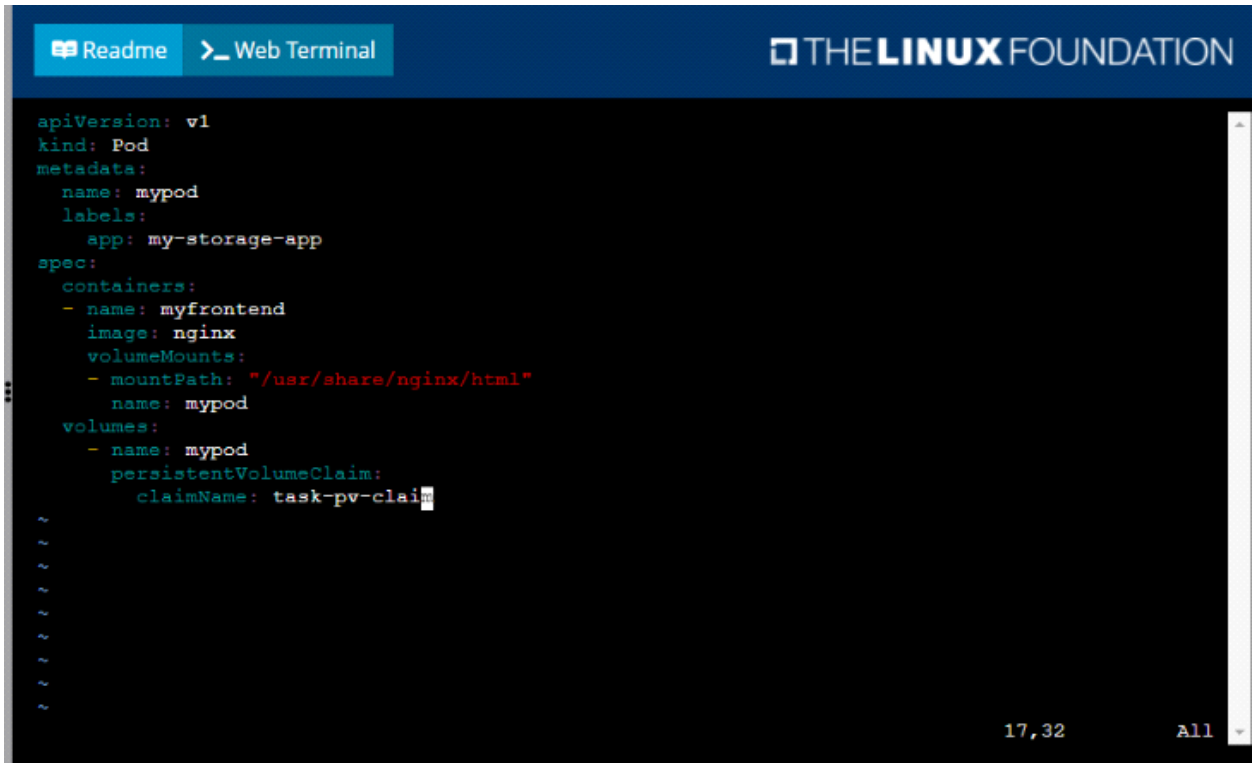
```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Oct  9 08:52:09 UTC 2020

  System load:  2.02                Users logged in:        0
  Usage of /:   10.3% of 242.29GB   IP address for eth0:    10.250.3.115
  Memory usage: 2%                  IP address for docker0: 172.17.0.1
  Swap usage:   0%                  IP address for 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:~$
```


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

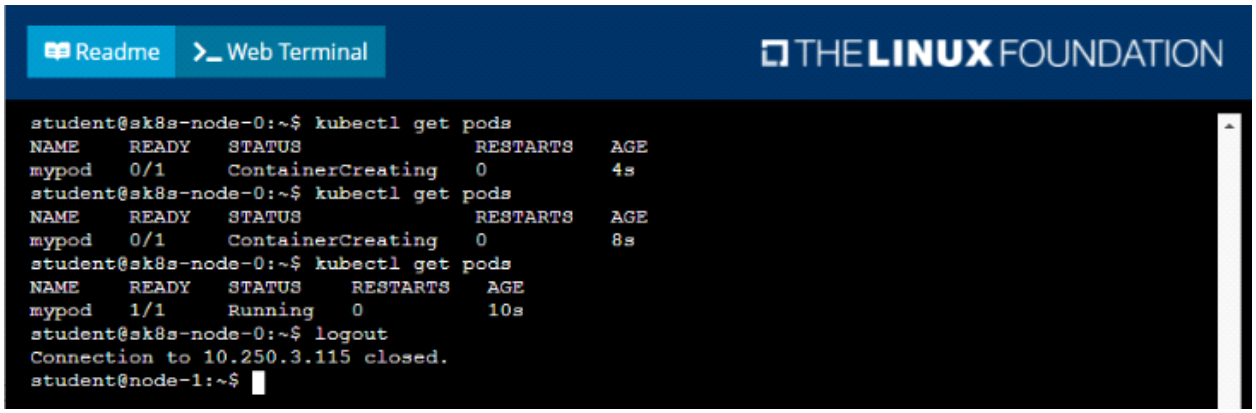

Readme
Web Terminal
THE LINUX FOUNDATION
-- INSERT --
0,1
All


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

```
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                   STO
RAGECLASS   REASON   AGE
task-pv-volume   1Gi        RWO            Retain           Bound    default/task-pv-claim   sto
rage                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
```




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



Hide Answer

Suggested Answer: *A*