Linux Foundation

CKAD

Certified Kubernetes Application Developer (CKAD) Program

Version: Demo

[Total Questions: 10]

Web: www.examout.co

Email: support@examout.co

IMPORTANT NOTICE

Feedback

We have developed quality product and state-of-art service to ensure our customers interest. If you have any suggestions, please feel free to contact us at feedback@examout.co

Support

If you have any questions about our product, please provide the following items:

- exam code
- screenshot of the question
- login id/email

please contact us at support@examout.co and our technical experts will provide support within 24 hours.

Copyright

Question #:1

Set configuration context:

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

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, usingCOOL_VARIABLE as the name for the environment variable inside the pod

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
```

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:~$
```

Question #:2

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 of200m CPU and1Gi memory for its container

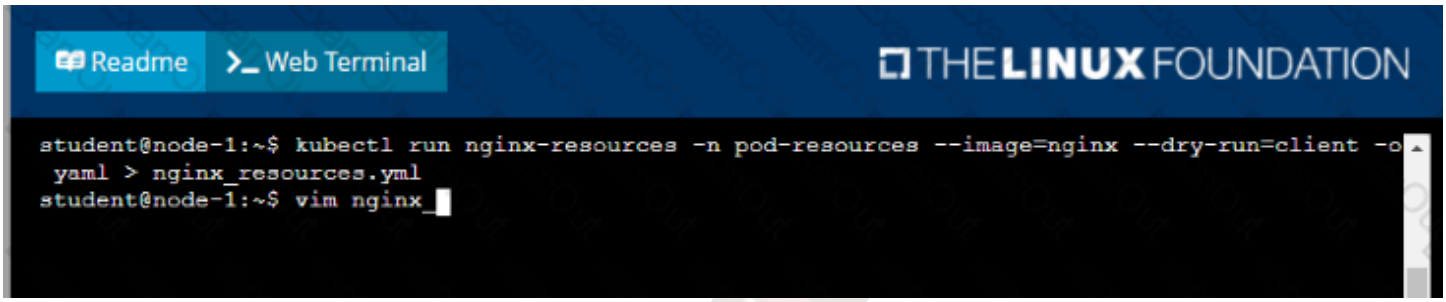
• The pod should use the nginx image

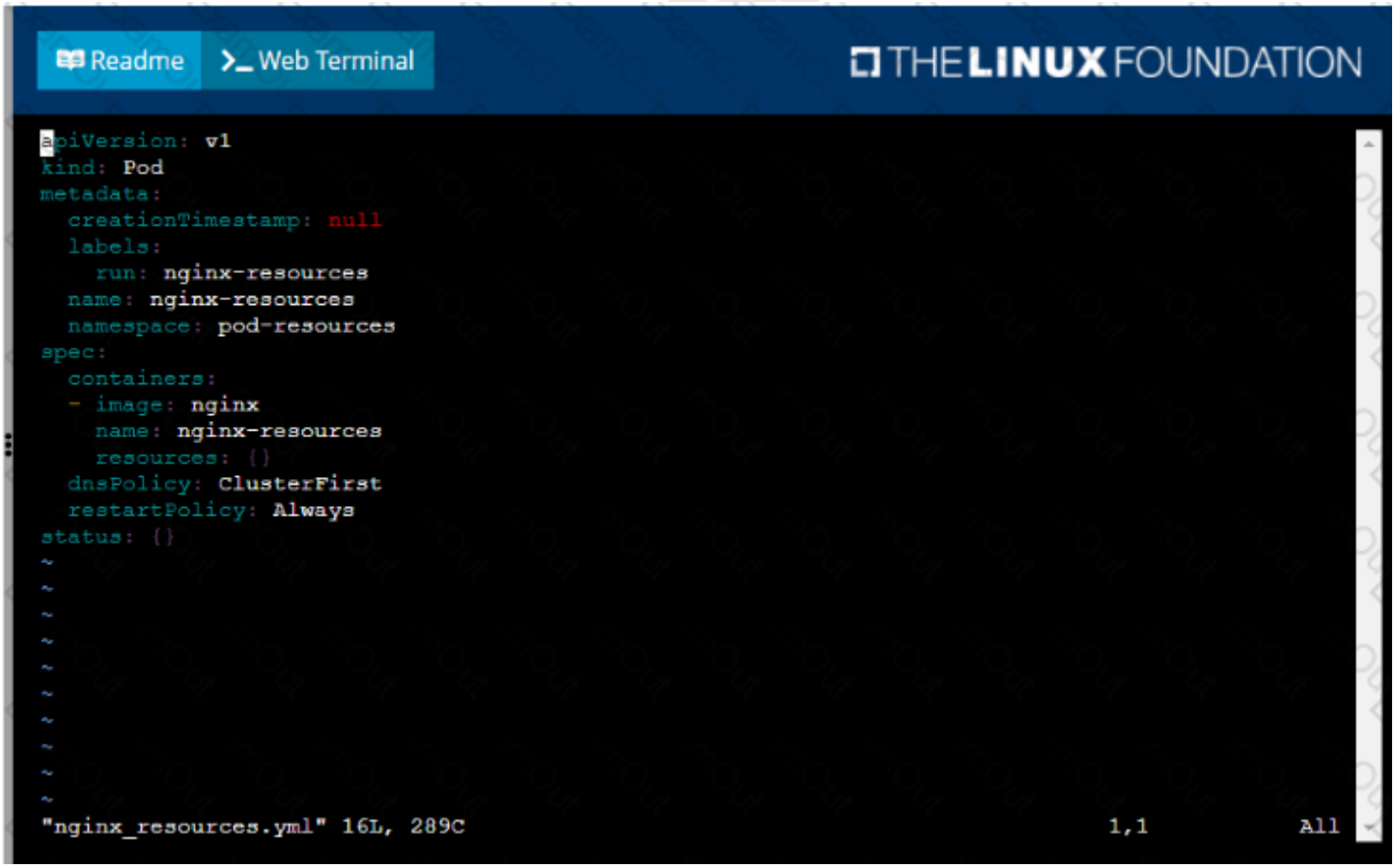
• The pod-resources namespace has already been created

See the solution below.

Explanation

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:~$
```

Question #:3

Set configuration context:

```
[student@node-1] $ kubectl config
use-context dk8s
```

Context

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

Task

Perform the following tasks:

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

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

The output file has already been created

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

• Fix the issue.

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

- qa
- test
- production
- alan

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 id86849c15382e; Security:[seccomp=unconfined]

23s 23s 1{kubelet worker0} spec.containers{liveness} **Normal**Started Started container with docker id86849c15382e

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 id86849c15382e; Security:[seccomp=unconfined]

36s 36s 1{kubelet worker0} spec.containers{liveness} Normal Started Started container**with**docker id86849c15382e

2s 2s 1{kubelet worker0} spec.containers{liveness} Warning Unhealthy Liveness probe failed: cat: can't open '/tmp/healthy': No such**fileor**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:

NAMEREADY **STATUS**RESTARTS AGE

liveness-exec 1/1Running 1m

Question #:4

Set configuration context:

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

Context

A pod is running on the cluster but it is not responding.

Task

The desired behavior is to have 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

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;sleep30;rm-rf/tmp/healthy;sleep600

livenessProbe:

exec:

command:

- cat

- /tmp/healthy

initialDelaySeconds:5

periodSeconds:5

In the configuration file, you can see that the Pod has a single Container. The periodSeconds field specifies that the kubelet should perform a liveness probe every 5 seconds. The initialDelaySeconds field tells the kubelet that it should wait 5 seconds before performing the first probe. To perform a probe, the kubelet executes the command cat /tmp/healthy in the target container. If the command succeeds, it returns 0, and the kubelet considers the container to be alive and healthy. If the command returns a non-zero value, the kubelet kills the container and restarts it.

When the container starts, it executes this command:

/bin/sh -c"touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600"

For the first 30 seconds of the container's life, there is a /tmp/healthy file. So during the first 30 seconds, the command cat /tmp/healthy returns a success code. After 30 seconds, cat /tmp/healthy returns a failure code.

Create the Pod:

kubectl apply -f https://k8s.io/examples/pods/probe/exec-liveness.yaml

Within 30 seconds, view the Pod events:

kubectl describe pod liveness-exec

The output indicates that no liveness probes have failed yet:

FirstSeen LastSeen Count From SubobjectPath Type Reason Message

--- --- --- --- --- --- --- ---

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

23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image "k8s.gcr.io/busybox"

23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image "k8s.gcr.io/busybox"

23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Created Created container with docker id 86849c15382e; Security:[seccomp=unconfined]

23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Started Started container with docker id 86849c15382e

After 35 seconds, view the Pod events again:

kubectl describe pod liveness-exec

At the bottom of the output, there are messages indicating that the liveness probes have failed, and the containers have been killed and recreated.

FirstSeen LastSeen Count From SubobjectPath Type Reason Message

--- --- --- --- --- --- --- ---

37s 37s 1 {default-scheduler } Normal Scheduled Successfully assigned liveness-exec to worker0

36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image "k8s.gcr.io/busybox"

36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image "k8s.gcr.io/busybox"

36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Created Created container with docker id 86849c15382e; Security:[seccomp=unconfined]

36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Started Started container with docker id 86849c15382e

2s 2s 1 {kubelet worker0} spec.containers{liveness} Warning Unhealthy Liveness probe failed: cat: can't open '/tmp/healthy': No such file or directory

Wait another 30 seconds, and verify that the container has been restarted:

kubectl get pod liveness-exec

The output shows that RESTARTS has been incremented:

NAME READY STATUS RESTARTS AGE

liveness-exec 1/1 Running 1 1m

Question #:5

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 with4replicas

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

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

See the solution below.

Explanation

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
```

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:~$
```

Question #:6

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

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

See the solution below.

Explanation

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","-B"]
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                   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

```
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 >
```

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:~$
```

Question #:7

Set configuration context:

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

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=webFrontEndkey/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

See the solution below.

Explanation

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
```

Question #:8

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 namedanother-config containing the key/value pair: key4/value3

• starta 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

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 > 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:~$
```

```
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:~$ 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
```

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
```

Question #:9

Set configuration context:

```
[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.

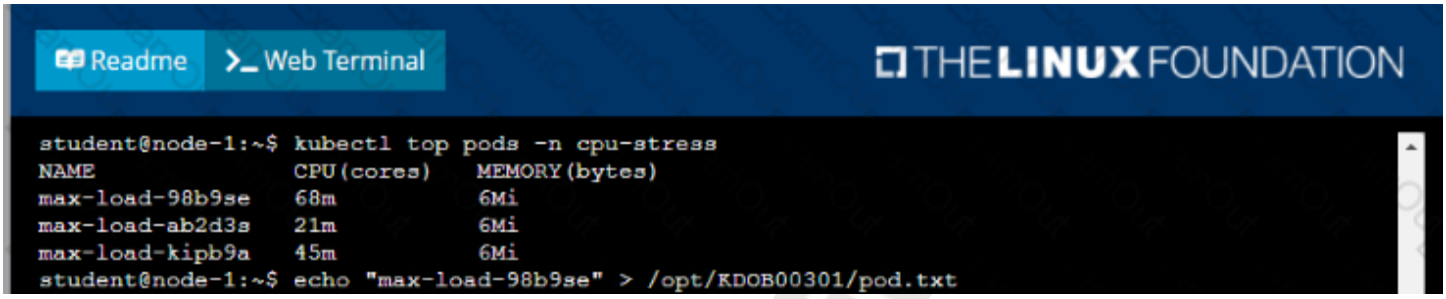
Task

• From the pods running in namespacecpu-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.

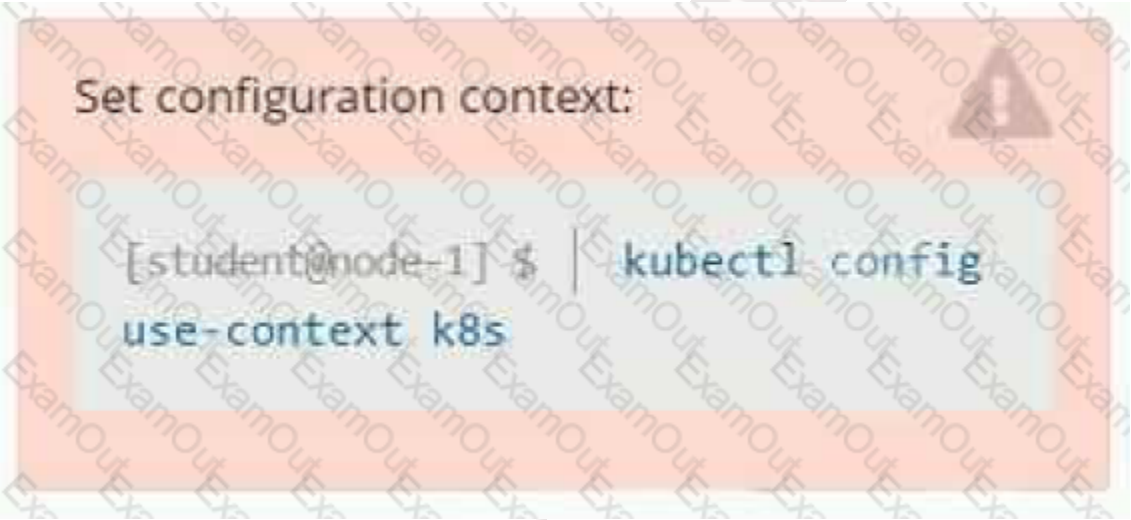
See the solution below.

Explanation

Solution:



Question #:10



Context

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

Task

Please complete the following:

• Update theappdeployment in the kdpd00202 namespace with a maxSurge of5% and a maxUnavailable of2%

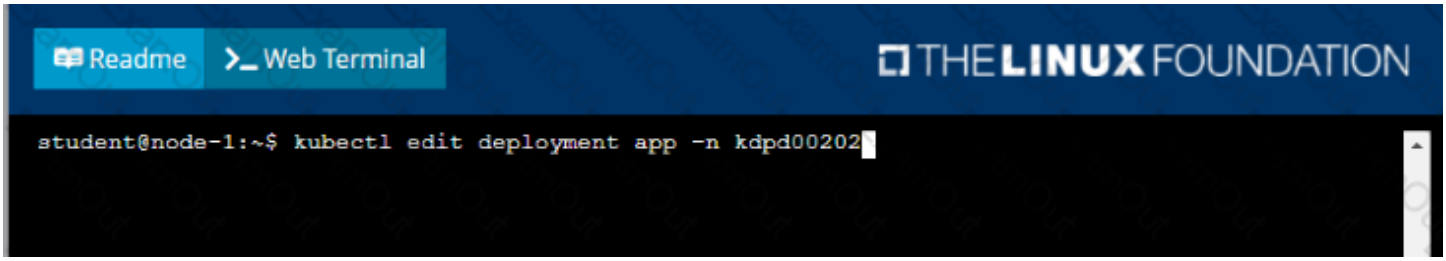
• Perform a rolling update of the web1 deployment, changing the Ifccncf/ngmx image version to 1.13

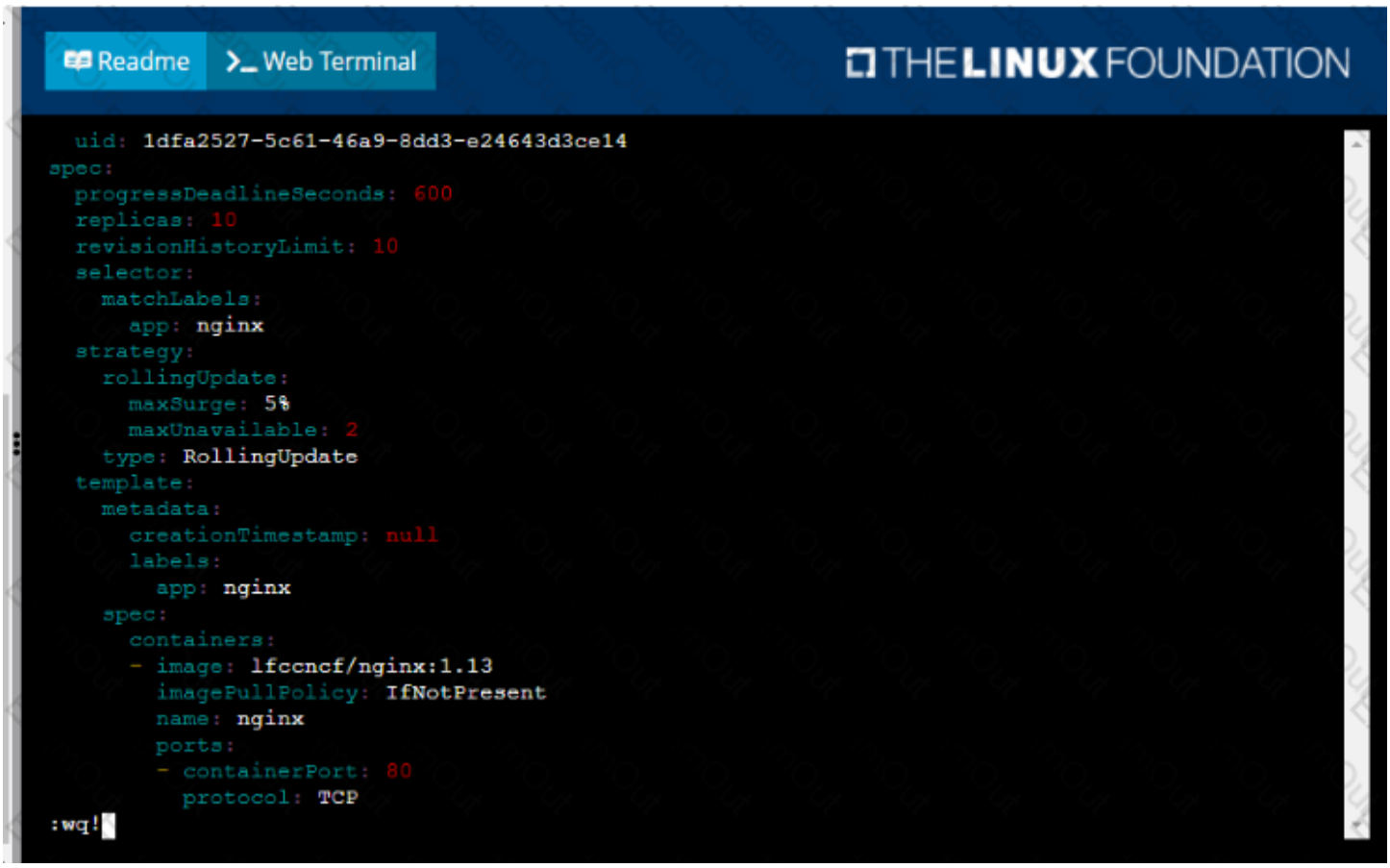
• Roll back theappdeployment to the previous version

See the solution below.

Explanation

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
```

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

About examout.co

examout.co was founded in 2007. We provide latest & high quality IT / Business Certification Training Exam Questions, Study Guides, Practice Tests.

We help you pass any IT / Business Certification Exams with 100% Pass Guaranteed or Full Refund. Especially Cisco, CompTIA, Citrix, EMC, HP, Oracle, VMware, Juniper, Check Point, LPI, Nortel, EXIN and so on.

View list of all certification exams: All vendors

We prepare state-of-the art practice tests for certification exams. You can reach us at any of the email addresses listed below.

- Sales: sales@examout.co
- Feedback: feedback@examout.co
- Support: support@examout.co

Any problems about IT certification or our products, You can write us back and we will get back to you within 24 hours.