# Newest CKAD Exam Topic - How to Download for Dump CKAD Collection Free of Charge

BONUS!!! Download part of DumpsTests CKAD dumps for free: https://drive.google.com/open?id=1DKC4P89v-7K7sYaoMQQ5TdYclxGWyp36

Though there are three versions of our CKAD exam braindumps: the PDF, Software and APP online. When using the APP version for the first time, you need to ensure that the network is unblocked, and then our CKAD guide questions will be automatically cached. The network is no longer needed the next time you use it. You can choose any version of our CKAD Practice Engine that best suits your situation. It's all for you to learn better.

The CKAD Exam is designed to help developers demonstrate their expertise in Kubernetes application development and gain recognition in the industry. Linux Foundation Certified Kubernetes Application Developer Exam certification is ideal for developers who work with Kubernetes on a daily basis and want to showcase their skills to potential employers. By passing the CKAD Exam, developers can prove their proficiency in building and managing cloud-native applications using Kubernetes and enhance their career prospects.

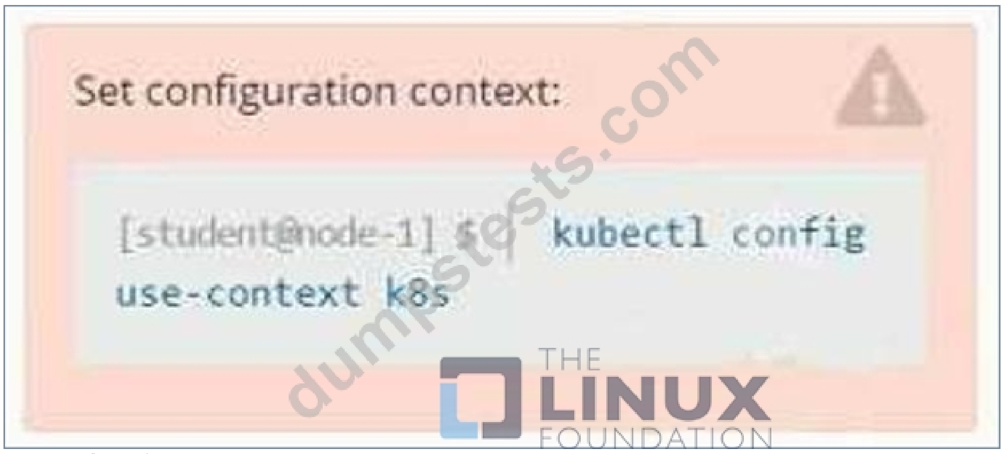
**>> CKAD Exam Topic <<**

## Practice Exam Software Linux Foundation CKAD Dumps PDF

DumpsTests You can modify settings of practice test in terms of Linux Foundation Certified Kubernetes Application Developer Exam CKAD Practice Questions types and mock exam duration. Both CKAD exam practice tests (web-based and desktop) save your every attempt and present result of the attempt on the spot. Actual exam environments of web-based and desktop Linux Foundation practice test help you overcome exam fear. Our Linux Foundation desktop practice test software works after installation on Windows computers.

## Linux Foundation Certified Kubernetes Application Developer Exam Sample Questions (Q68-Q73):

**NEW QUESTION # 68**
Refer to Exhibit.

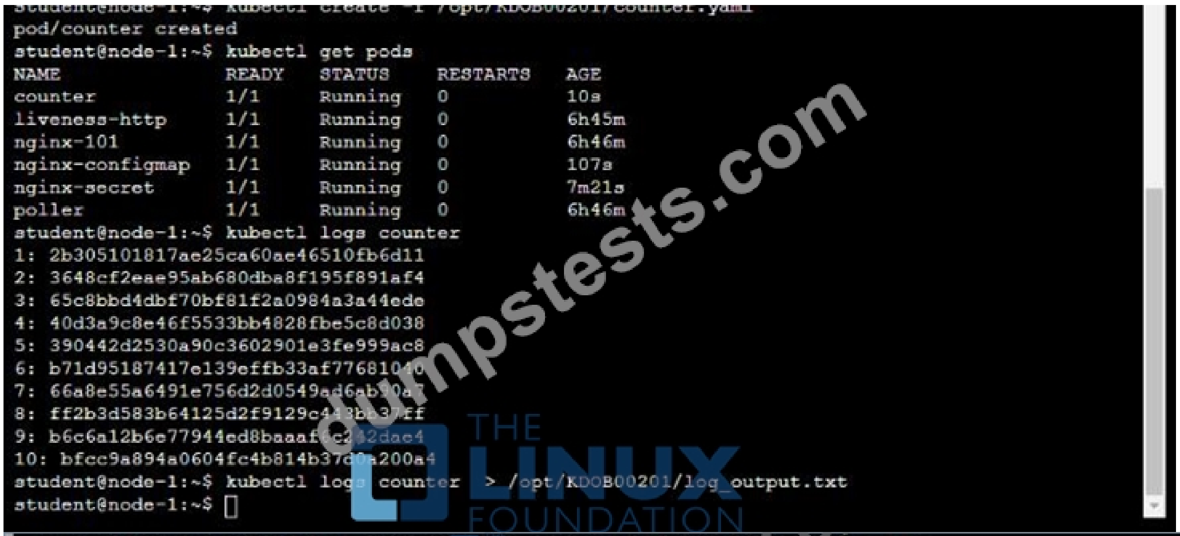

Set configuration context:

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

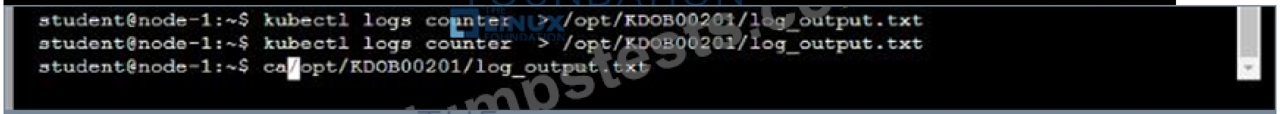
Set Configuration Context:
[student@node-1] $ | kubectl
Config use-context k8s
Context
You sometimes need to observe a pod's logs, and write those logs to a file for further analysis.
Task
Please complete the following;
* Deploy the counter pod to the cluster using the provided YAMLspec file at /opt/KDOB00201/counter.yaml
* Retrieve all currently available application logs from the running pod and store them in the file /opt/KDOB0020l/log_Output.txt, which has already been created

**Answer:**

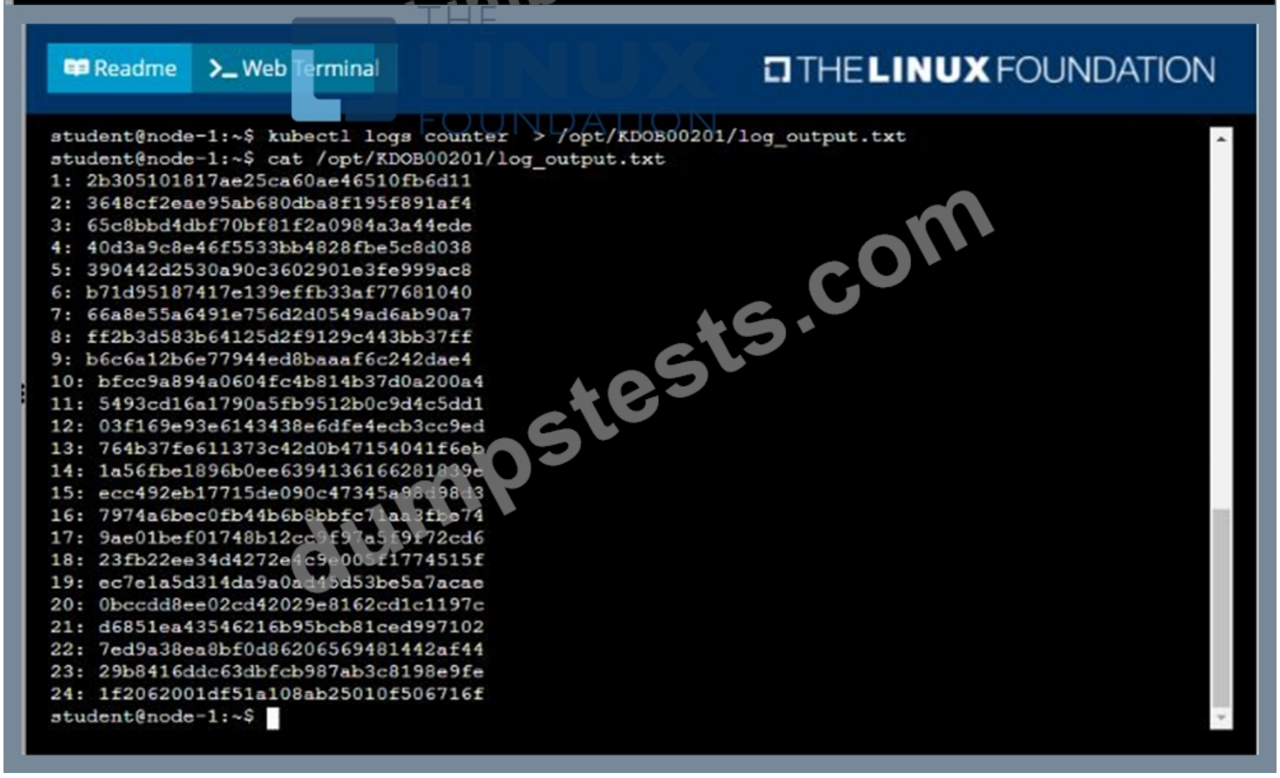
Explanation:
Solution:
To deploy the counter pod to the cluster using the provided YAML spec file, you can use the kubectl apply command. The apply command creates and updates resources in a cluster.
kubectl apply -f /opt/KDOB00201/counter.yaml
This command will create the pod in the cluster. You can use the kubectl get pods command to check the status of the pod and ensure that it is running.
kubectl get pods
To retrieve all currently available application logs from the running pod and store them in the file /opt/KDOB0020l/log_Output.txt, you can use the kubectl logs command. The logs command retrieves logs from a container in a pod.
kubectl logs -f <pod-name> > /opt/KDOB0020l/log_Output.txt
Replace <pod-name> with the name of the pod.
You can also use -f option to stream the logs.
kubectl logs -f <pod-name> > /opt/KDOB0020l/log_Output.txt &
This command will retrieve the logs from the pod and write them to the /opt/KDOB0020l/log_Output.txt file.
Please note that the above command will retrieve all logs from the pod, including previous logs. If you want to retrieve only the new logs that are generated after running the command, you can add the --since flag to the kubectl logs command and specify a duration, for example --since=24h for logs generated in the last 24 hours.
Also, please note that, if the pod has multiple containers, you need to specify the container name using -c option.
kubectl logs -f <pod-name> -c <container-name> > /opt/KDOB0020l/log_Output.txt The above command will redirect the logs of the specified container to the file.



```
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:~$ 
```



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



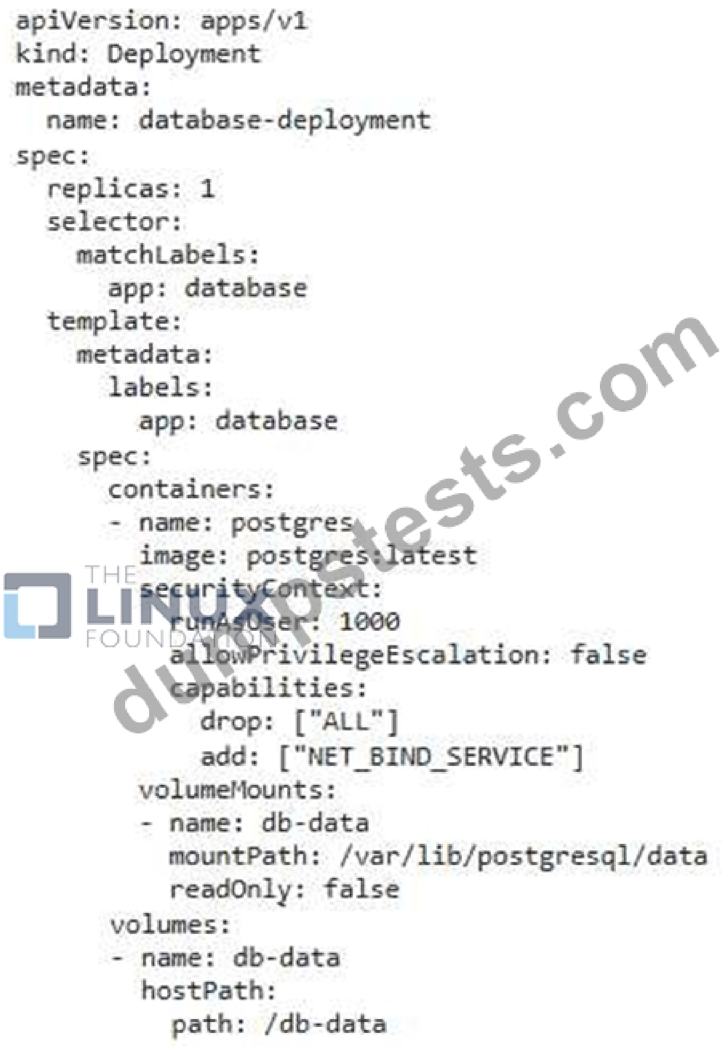
```
student@node-1:~$ kubectl logs counter  > /opt/KDOB00201/log_output.txt
student@node-1:~$ cat /opt/KDOB00201/log_output.txt
1: 2b305101817ae25ca60ae46510fb6d11
2: 3648cf2eae95ab680dba8f195f891af4
3: 65c8bbd4dbf70bf81f2a0984a3a44ede
4: 40d3a9c8e46f5533bb4828fbe5c8d038
5: 390442d2530a90c3602901e3fe999ac8
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:~$ 
```

**NEW QUESTION # 69**
You have a Deployment named 'database-deployment' that runs a PostgreSQL database container. You want to enforce the following security restrictions:
- The container should only be allowed to run with the I-IID 1000.
- The container should be able to access a specific hostPath volume mounted at '/db-data' for storing database data.
- The container should not be allowed to escalate privileges.
- The container should only have the 'NET BIND SERVICE capability, allowing it to listen on specific ports.
You need to define a SecurityContext in the Deployment configuration to enforce these restrictions.

**Answer:**

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :

1. Define the SecurityContext
- Create a 'securitycontext' section within the 'spec-template-spec-containers' block for your 'database-deployment container-
- Set 'runAsIJsers to '1000' to enforce running as UID 1000.
- Set 'allowPrivilegeEscalation' to 'false' to disable privilege escalation-
- In the 'capabilities' sectiom
- Set 'drop' to an array containing all capabilities except 'NET BIND_SERVICE'
- Set 'add' to an array containing 'NET BIND SERVICE
- Define a 'volumeMount' to mount the '/db-data' hostPath volume.
Solution (Step by Step) :
1. Define the SecurityContext:
- Create a 'securityContext' section within the block for your 'database-deployment container.
- Set 'runAslJser' to ''1000' to enforce running as UID 1000.
- Set 'allowPriviIegeEscaIation' to 'false' to disable privilege escalation.
- In the 'capabilities' section:
- Set 'drop' to an array containing all capabilities except 'NET BIND SERVICE
- Set 'add' to an array containing
- Define a 'volumeMount' to mount the '/db-data' hostPath volume.

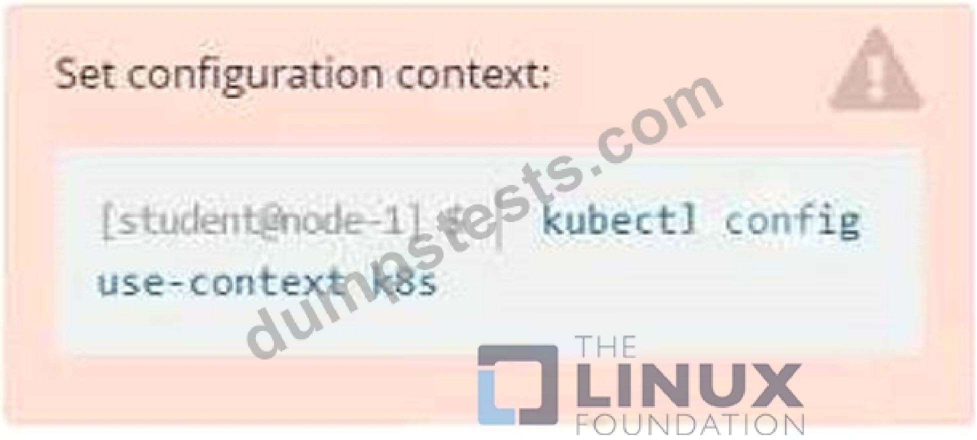


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: database-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: database
  template:
    metadata:
      labels:
        app: database
    spec:
      containers:
      - name: postgres
        image: postgres:latest
        securityContext:
          runAsUser: 1000
          allowPrivilegeEscalation: false
          capabilities:
            drop: ["ALL"]
            add: ["NET_BIND_SERVICE"]
        volumeMounts:
        - name: db-data
          mountPath: /var/lib/postgresql/data
          readOnly: false
      volumes:
      - name: db-data
        hostPath:
          path: /db-data
```

2. Create the Deployment: - Apply the Deployment YAML file using 'kubectl apply -f database-deployment.yaml. - The 'securityContext' restricts the container's benavior and capabilities. - Setting 'runAslJser' to '1000' forces the container to run as the specified UID. - 'allowPrivilegeEscalation' set to 'false' prevents tne container from gaining higner privileges. - The 'capabilities' section controls specific capabilities. 'drop' removes unwanted capabilities, while 'add' grants specific capabilities. In this case, the container is allowed to use the capability, enabling it to bind to specific ports. - The 'volumeMount defines the mount point for the hostPath volume, providing access to the specified directory tor database data. This configuration ensures that the 'database-deployment container runs with the specific IJID, cannot escalate privileges, and only has the 'NET BIND SERVICE' capability, while accessing the hostPath volume for database data. This provides a secure environment for your database container.,

**NEW QUESTION # 70**

Context



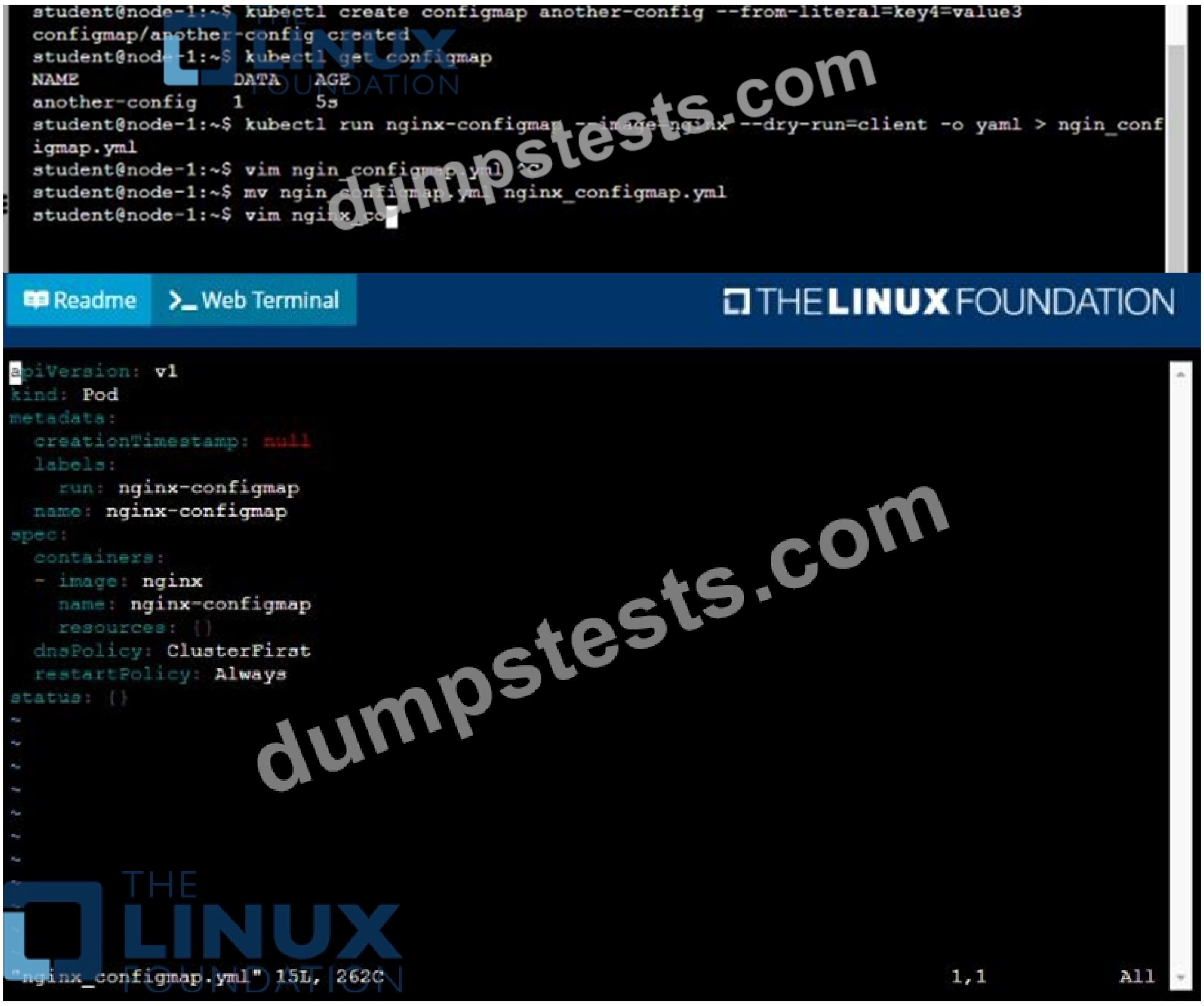
Set configuration context:

```
[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

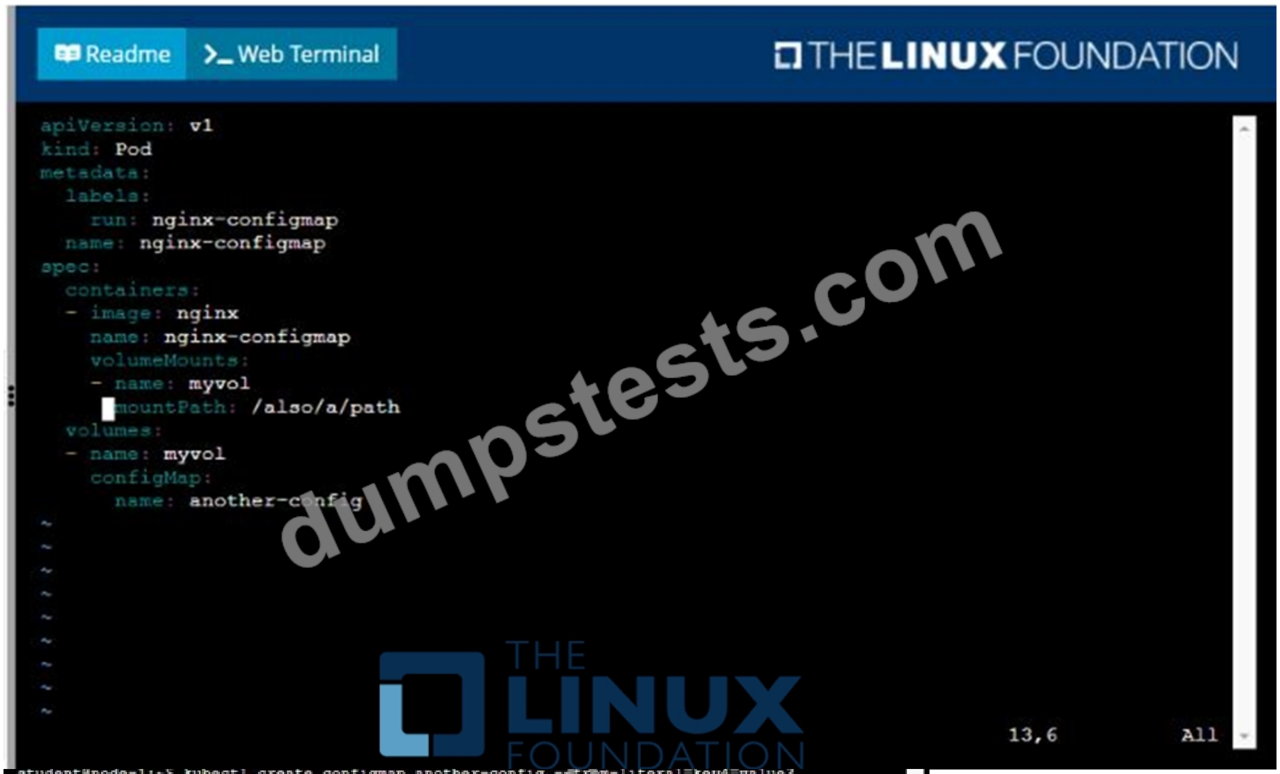
**Answer:**

Explanation:
Solution:

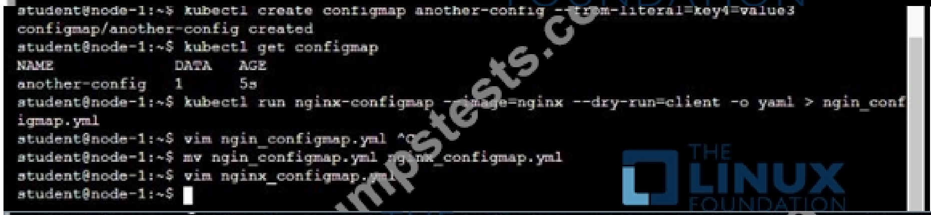


```
student@node-1:~$ kubectl create configmap another-config --from-literal=key4=value3
configmap/another-config created
student@node-1:~$ kubectl get configmap
NAME             DATA   AGE
another-config   1      5s
student@node-1:~$ kubectl run nginx-configmap --image=nginx --dry-run=client -o yaml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

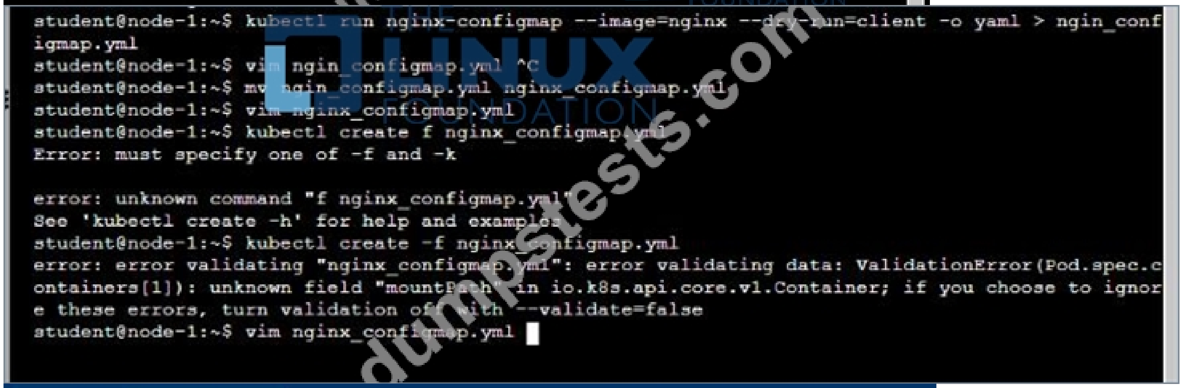
```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx-configmap
  name: nginx-configmap
spec:
  containers:
  - image: nginx
    name: nginx-configmap
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
~
~
~
~
~
~
~
~
~
"nginx_configmap.yml" 15L, 262C                                1,1           All
```



```
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
```

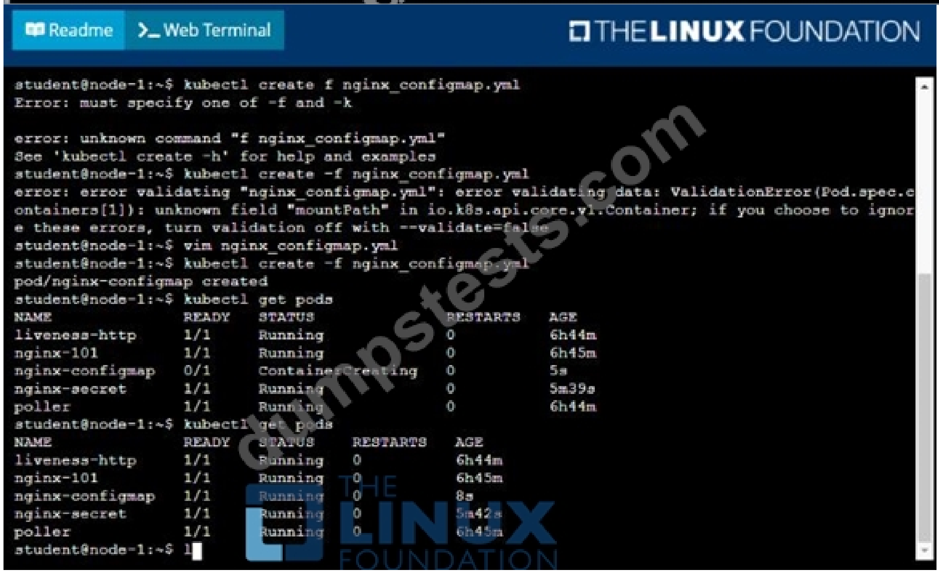


```
student@node-1:~$ kubectl create configmap another-config --from-literal=key4=value3
configmap/another-config created
student@node-1:~$ kubectl get configmap
NAME             DATA   AGE
another-config   1      5s
student@node-1:~$ kubectl run nginx-configmap --image=nginx --dry-run=client -o yaml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



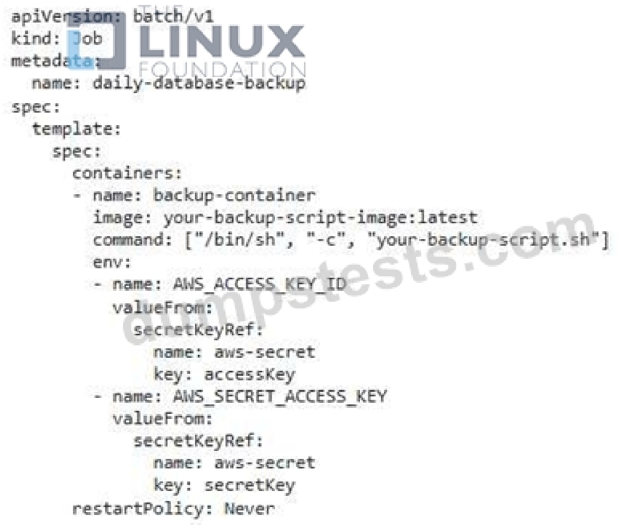
```
student@node-1:~$ kubectl create f nginx_configmap.yml
Error: must specify one of -f and -k

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
```

**NEW QUESTION # 71**
You are building a system for scheduling daily backups of a critical database. The backup process involves running a script that connects to the database, extracts tne data, and saves it to an S3 bucket. How would you utilize Kubernetes JobS to automate this backup process and ensure it runs every day at 2:00 AM?

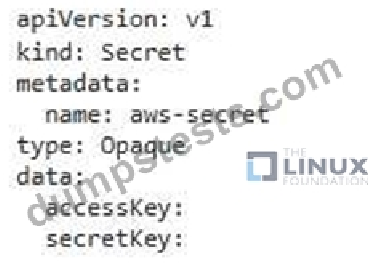
**Answer:**

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create a Job YAML file.



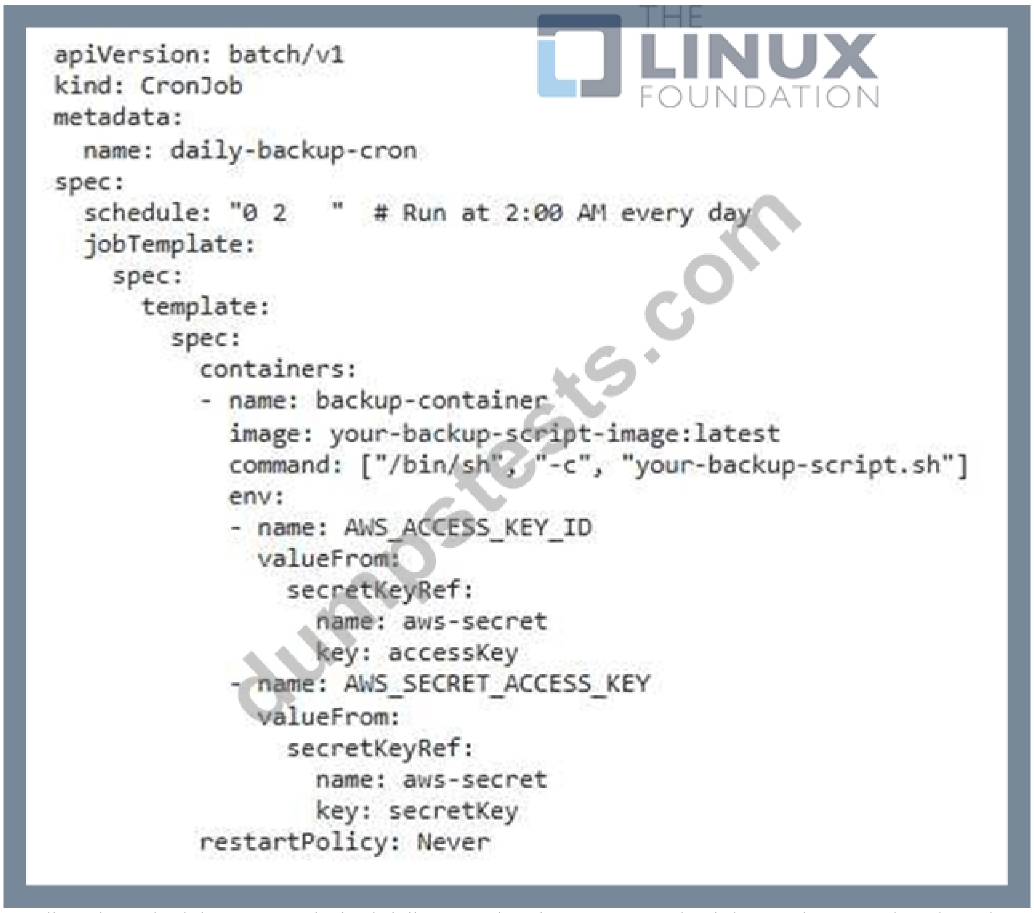
```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: daily-database-backup
spec:
  template:
    spec:
      containers:
      - name: backup-container
        image: your-backup-script-image:latest
        command: ["/bin/sh", "-c", "your-backup-script.sh"]
        env:
        - name: AWS_ACCESS_KEY_ID
          valueFrom:
            secretKeyRef:
              name: aws-secret
              key: accessKey
        - name: AWS_SECRET_ACCESS_KEY
          valueFrom:
            secretKeyRef:
              name: aws-secret
              key: secretKey
      restartPolicy: Never
```

- Replace 'your-backup-script-image:latest With the actual image name of your backup script. - Replace 'your-backup-script.sn' with the actual name Of your backup script. - Replace saws-secret' with the name of the Kubernetes secret holding your AWS credentials (see step 2). - 'restartPolicy: Never ensures the job runs only once. 2. Create a Secret for AWS Credentials:



```yaml
apiVersion: v1
kind: Secret
metadata:
  name: aws-secret
type: Opaque
data:
  accessKey:
  secretKey:
```

- Replace "and" With your actual AWS credentials. 3. Create a CronJob YAML file:

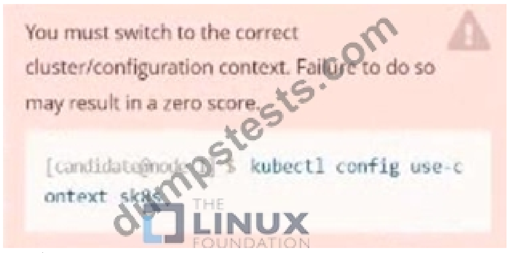


```yaml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: daily-backup-cron
spec:
  schedule: "0 2   "  # Run at 2:00 AM every day
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: backup-container
            image: your-backup-script-image:latest
            command: ["/bin/sh", "-c", "your-backup-script.sh"]
            env:
            - name: AWS_ACCESS_KEY_ID
              valueFrom:
                secretKeyRef:
                  name: aws-secret
                  key: accessKey
            - name: AWS_SECRET_ACCESS_KEY
              valueFrom:
                secretKeyRef:
                  name: aws-secret
                  key: secretKey
          restartPolicy: Never
```

- Adjust the 'schedules to your desired daily execution time. - Ensure the 'jobTemplate' matches the Job YAML definition. 4. Apply the YAML files: - Use 'kubectl apply -f job.yamr and 'kubectl apply -f cronjob.yamr to create the Job and CronJob on your cluster 5. Verify the CronJob: - Use ' kubectl get cronjobs' to check the status of the CronJob- - You should see the CronJob running and triggering the Job at the specified time.

**NEW QUESTION # 72**

Context



You must switch to the correct cluster/configuration context. Failure to do so may result in a zero score.

```
[candidate@node-1] $ kubectl config use-c
ontext sk8s
```

Task:

1) First update the Deployment cka00017-deployment in the ckad00017 namespace:

To run 2 replicas of the pod

Add the following label on the pod:

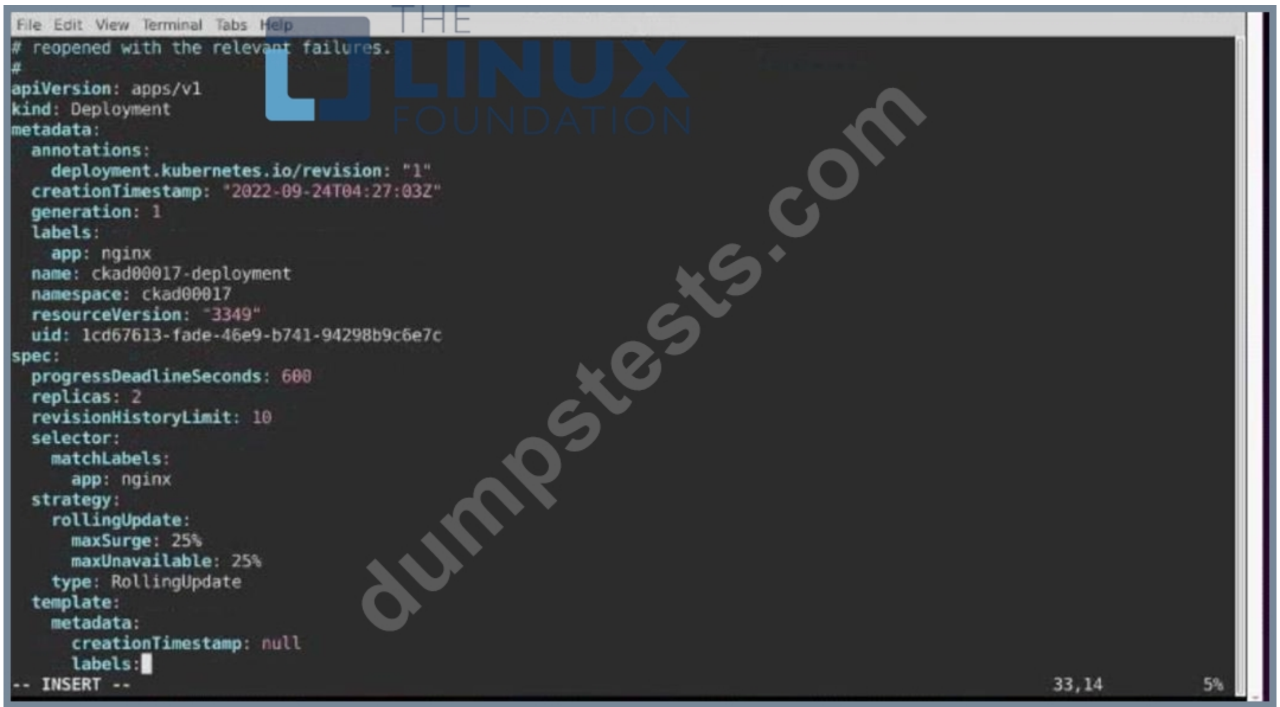
Role userUI

2) Next, Create a NodePort Service named cherry in the ckad00017 nmespace exposing the ckad00017-deployment Deployment on TCP port 8888

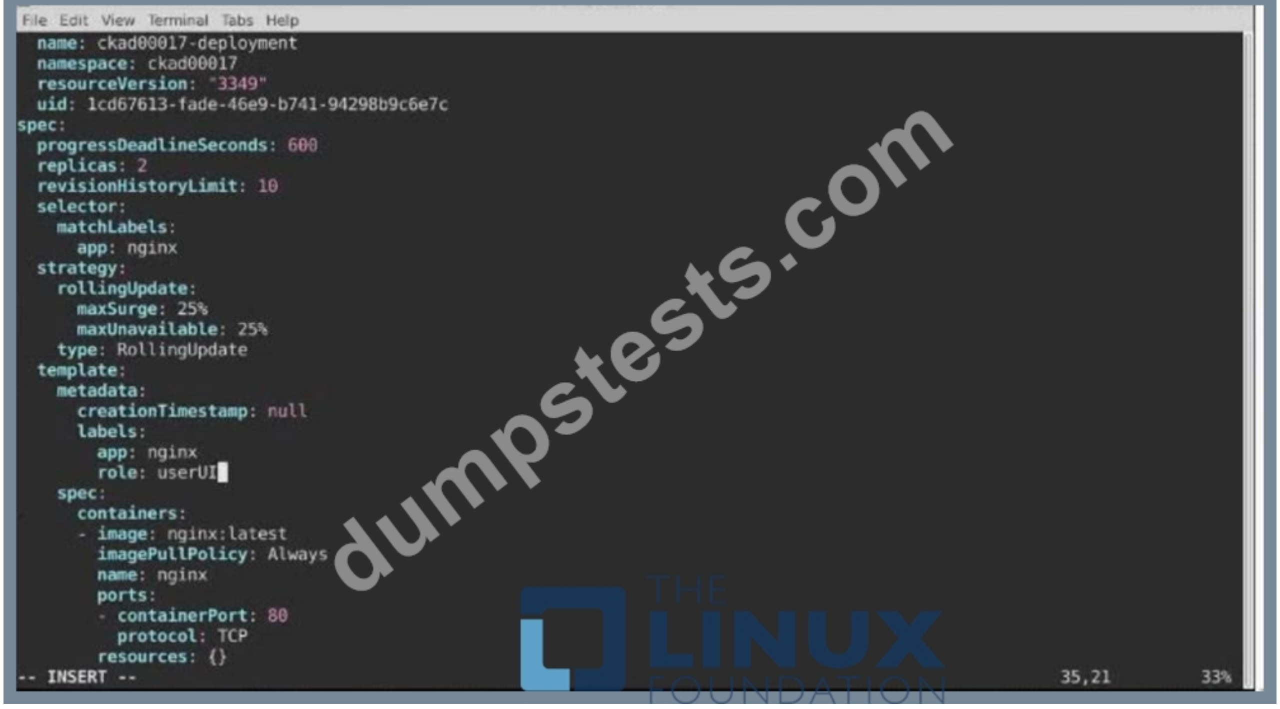
**Answer:**

Explanation:

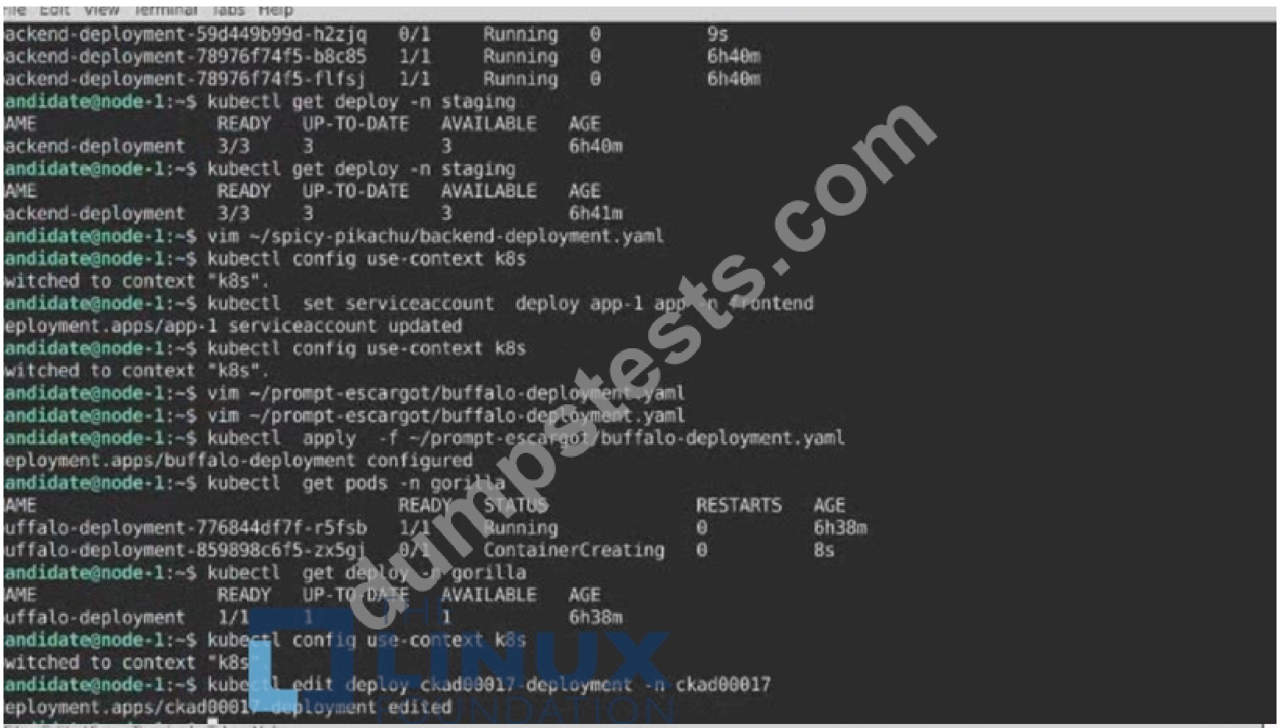
Solution:



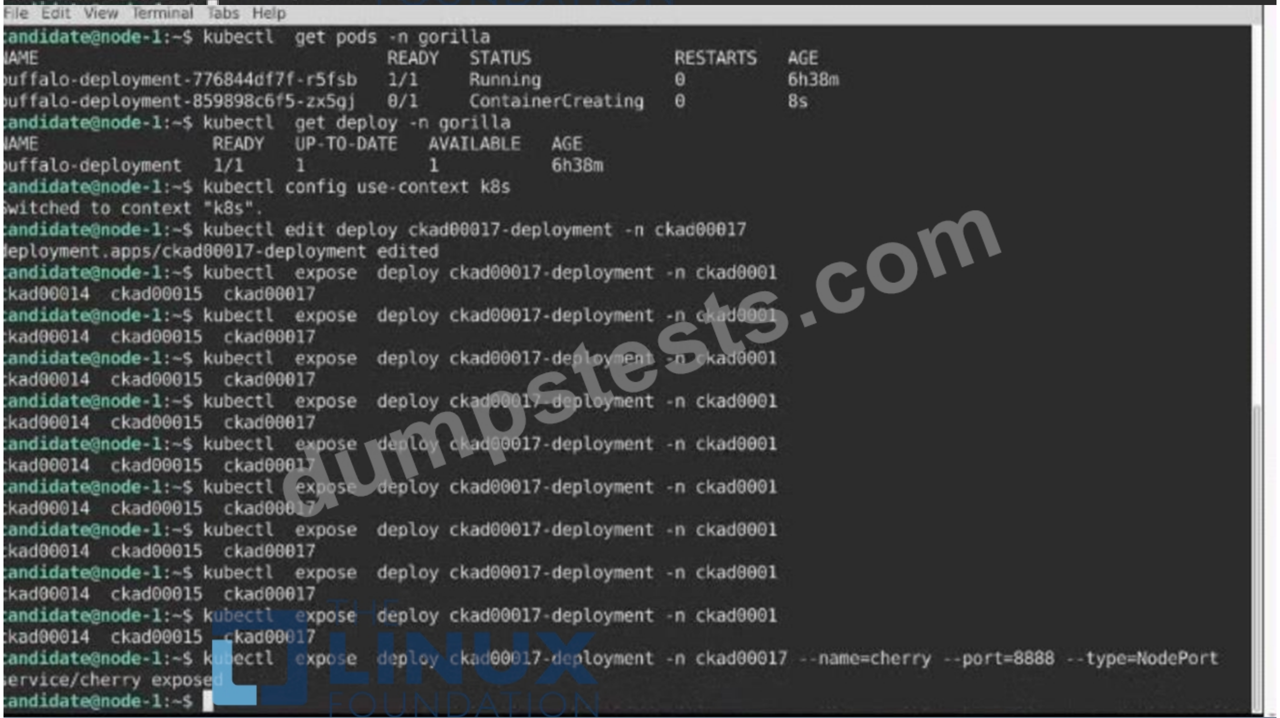
```
# reopened with the relevant failures.
#
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: "2022-09-24T04:27:03Z"
  generation: 1
  labels:
    app: nginx
  name: ckad00017-deployment
  namespace: ckad00017
  resourceVersion: "3349"
  uid: 1cd67613-fade-46e9-b741-94298b9c6e7c
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
-- INSERT --                                                     33,14         5%
```



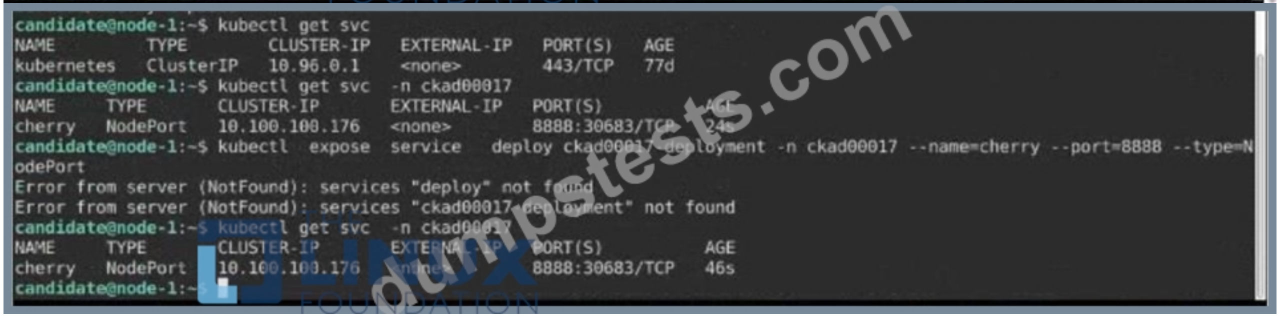
```
  name: ckad00017-deployment
  namespace: ckad00017
  resourceVersion: "3349"
  uid: 1cd67613-fade-46e9-b741-94298b9c6e7c
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
        role: userUI
    spec:
      containers:
      - image: nginx:latest
        imagePullPolicy: Always
        name: nginx
        ports:
        - containerPort: 80
          protocol: TCP
        resources: {}
-- INSERT --                                                     35,21        33%
```



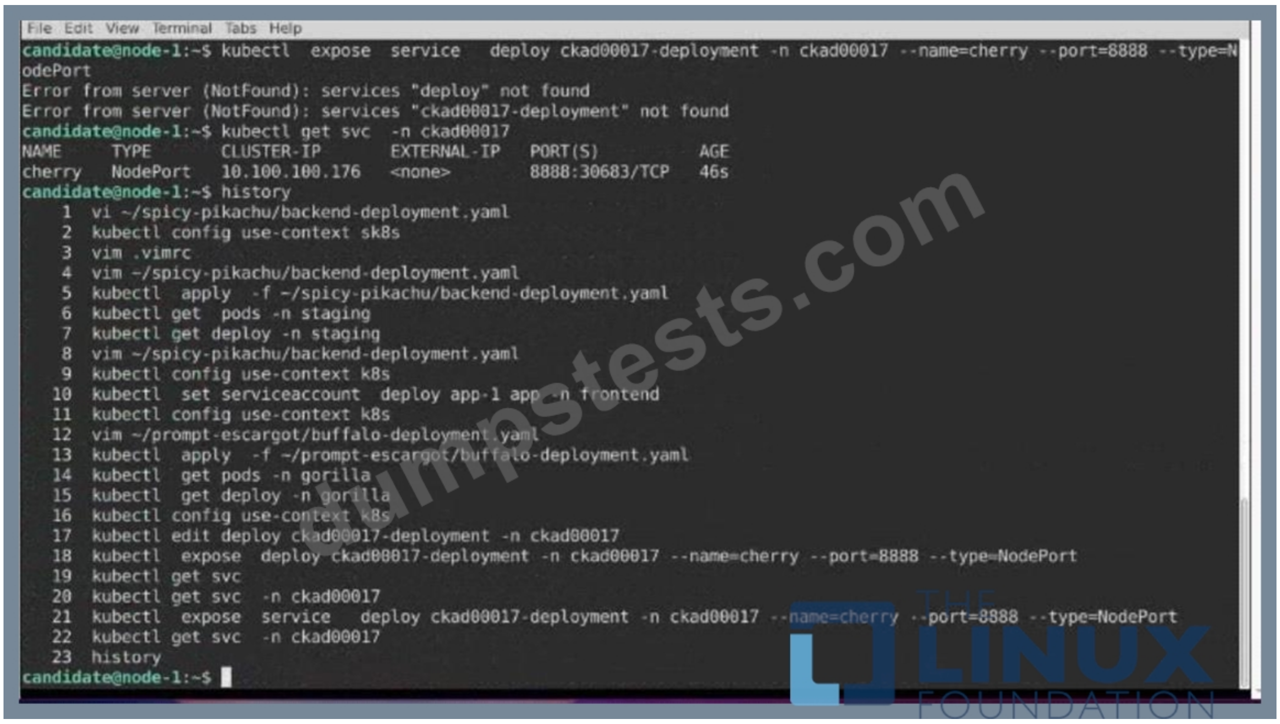
```
ackend-deployment-59d449b99d-h2zjq   0/1     Running   0          9s
ackend-deployment-78976f74f5-b8c85   1/1     Running   0          6h40m
ackend-deployment-78976f74f5-flfsj   1/1     Running   0          6h40m
andidate@node-1:~$ kubectl get deploy -n staging
AME                  READY   UP-TO-DATE   AVAILABLE   AGE
ackend-deployment    3/3     3            3           6h40m
andidate@node-1:~$ kubectl get deploy -n staging
AME                  READY   UP-TO-DATE   AVAILABLE   AGE
ackend-deployment    3/3     3            3           6h41m
andidate@node-1:~$ vim ~/spicy-pikachu/backend-deployment.yaml
andidate@node-1:~$ kubectl config use-context k8s
witched to context "k8s".
andidate@node-1:~$ kubectl  set serviceaccount  deploy app-1 app -n frontend
eployment.apps/app-1 serviceaccount updated
andidate@node-1:~$ kubectl config use-context k8s
witched to context "k8s".
andidate@node-1:~$ vim ~/prompt-escargot/buffalo-deployment.yaml
andidate@node-1:~$ vim ~/prompt-escargot/buffalo-deployment.yaml
andidate@node-1:~$ kubectl  apply  -f ~/prompt-escargot/buffalo-deployment.yaml
eployment.apps/buffalo-deployment configured
andidate@node-1:~$ kubectl  get pods -n gorilla
AME                                  READY   STATUS              RESTARTS   AGE
uffalo-deployment-776844df7f-r5fsb   1/1     Running             0          6h38m
uffalo-deployment-859898c6f5-zx5gj   0/1     ContainerCreating   0          8s
andidate@node-1:~$ kubectl  get deploy -n gorilla
AME                  READY   UP-TO-DATE   AVAILABLE   AGE
uffalo-deployment    1/1     1            1           6h38m
andidate@node-1:~$ kubectl config use-context k8s
witched to context "k8s".
andidate@node-1:~$ kubectl edit deploy ckad00017-deployment -n ckad00017
eployment.apps/ckad00017-deployment edited
```



```
andidate@node-1:~$ kubectl  get pods -n gorilla
IAME                                  READY   STATUS              RESTARTS   AGE
ouffalo-deployment-776844df7f-r5fsb   1/1     Running             0          6h38m
ouffalo-deployment-859898c6f5-zx5gj   0/1     ContainerCreating   0          8s
andidate@node-1:~$ kubectl  get deploy -n gorilla
IAME                  READY   UP-TO-DATE   AVAILABLE   AGE
ouffalo-deployment    1/1     1            1           6h38m
andidate@node-1:~$ kubectl config use-context k8s
witched to context "k8s".
andidate@node-1:~$ kubectl edit deploy ckad00017-deployment -n ckad00017
leployment.apps/ckad00017-deployment edited
andidate@node-1:~$ kubectl  expose  deploy ckad00017-deployment -n ckad0001
kad00014  ckad00015  ckad00017
andidate@node-1:~$ kubectl  expose  deploy ckad00017-deployment -n ckad0001
kad00014  ckad00015  ckad00017
andidate@node-1:~$ kubectl  expose  deploy ckad00017-deployment -n ckad0001
kad00014  ckad00015  ckad00017
andidate@node-1:~$ kubectl  expose  deploy ckad00017-deployment -n ckad0001
kad00014  ckad00015  ckad00017
andidate@node-1:~$ kubectl  expose  deploy ckad00017-deployment -n ckad0001
kad00014  ckad00015  ckad00017
andidate@node-1:~$ kubectl  expose  deploy ckad00017-deployment -n ckad0001
kad00014  ckad00015  ckad00017
andidate@node-1:~$ kubectl  expose  deploy ckad00017-deployment -n ckad0001
kad00014  ckad00015  ckad00017
andidate@node-1:~$ kubectl  expose  deploy ckad00017-deployment -n ckad0001
kad00014  ckad00015  ckad00017
andidate@node-1:~$ kubectl  expose  deploy ckad00017-deployment -n ckad0001
kad00014  ckad00015  ckad00017
andidate@node-1:~$ kubectl  expose  deploy ckad00017-deployment -n ckad00017 --name=cherry --port=8888 --type=NodePort
ervice/cherry exposed
andidate@node-1:~$
```



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



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

**NEW QUESTION # 73**

......

Do not worry because Linux Foundation CKAD exams are here to provide you with the exceptional Linux Foundation CKAD Dumps exams. Linux Foundation CKAD dumps Questions will help you secure the Linux Foundation CKAD certificate on the first go. As stated above, Linux Foundation Certified Kubernetes Application Developer Exam resolve the issue the aspirants encounter of finding reliable and original certification Exam Questions.

**Dump CKAD Collection**: https://www.dumpstests.com/CKAD-latest-test-dumps.html

- study.stcs.edu.np, www.qianqi.cloud, www.stes.tyc.edu.tw, courses.solutionbhai.com, www.stes.tyc.edu.tw, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, mahiracademy.com, pct.edu.pk, motionentrance.edu.np, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, Disposable vapes