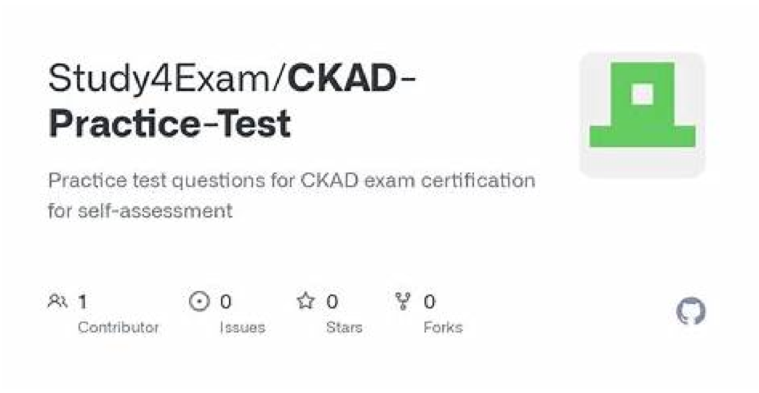
CKAD Practice Test Fee | CKAD Actualtest


Study4Exam/CKAD-Practice-Test
Practice test questions for CKAD exam certification for self-assessment
1 Contributor
0 Issues
0 Stars
0 Forks

P.S. Free 2025 Linux Foundation CKAD dumps are available on Google Drive shared by TestsDumps: https://drive.google.com/open?id=1n_cn2D9yQVh3-qiLjkp3Lq-v3qjdTWPb

Linux Foundation CKAD certification exam is very important for every IT person. With this certification you will not be eliminated, and you will be a raise. Some people say that to pass the Linux Foundation CKAD exam certification is tantamount to success. Yes, this is true. You get what you want is one of the manifestations of success. TestsDumps of Linux Foundation CKAD Exam Materials is the source of your success. With this training materials, you will speed up the pace of success, and you will be more confident.

To prepare for the exam, candidates can take advantage of a range of resources provided by the Linux Foundation, including online courses, practice exams, and study guides. They can also join online communities and participate in forums to connect with other developers who are preparing for the exam and share tips and strategies.

>> CKAD Practice Test Fee <<

CKAD Actualtest - CKAD Exam Details

In today's fast-paced world, having access to Linux Foundation Certified Kubernetes Application Developer Exam (CKAD) study material on the go is important. TestsDumps Linux Foundation Certified Kubernetes Application Developer Exam (CKAD) PDF questions are compatible with all smart devices, allowing you to study and prepare for the CKAD Exam whenever and wherever you choose. Since you can access real Linux Foundation CKAD dumps in PDF from your smartphone or tablet, you can easily fit CKAD exam preparation into your busy schedule.

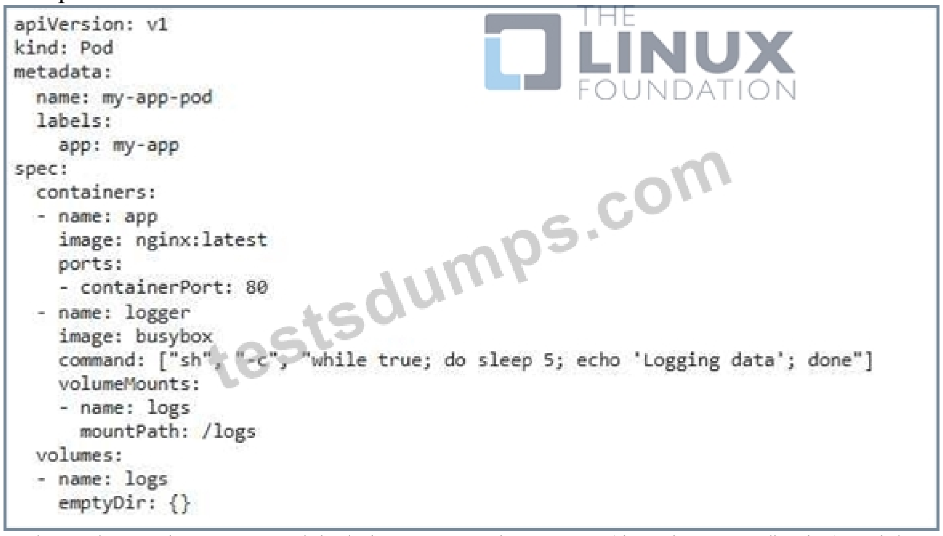
Linux Foundation Certified Kubernetes Application Developer Exam Sample Questions (Q25-Q30):

NEW QUESTION # 25
You are building a Kubernetes application that involves a microservice architecture with multiple pods for each service. One of your services requires a sidecar container to handle logging and monitoring. How would you design the pod structure and define the relationships between the application container and the sidecar container?

Answer:

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Define Pod Specification:
- Create a pod definition file (e.g., 'pod.yaml').
- Include the 'apiVersion', 'kind', 'metadata' (name, labels), and 'spec' sections.
2. Define Application Container:
- Within the 'spec.containerS section, define the primary application container:
- 'name': Provide a descriptive name for the application container (e.g., 'app').
- Simage: Specify the Docker image for the application.

- 'ports': Define any ports that the application exposes.
- 'resources': (Optional) Specify resource requests and limits for the application container.
3. Define Sidecar Container.
- Add another container definition within the 'spec-containers' section for the sidecar:
- 'name': Provide a name for the sidecar container (e.g., Slogger').
- 'image': Specify the Docker image for the sidecar container (e.g., "busybox'
- 'command': Define the command to run within the sidecar. This might involve using a logging agent, monitoring tool, or any other custom script.
- 'volumeMountss: (Optional) If the sidecar needs access to shared data, mount volumes here.
4. Define Shared Volumes (Optional):
- If necessary, create a 'spec-volumes' section to define any shared volumes that both containers can access. This might include:
- 'emptyDir': For temporary data that only exists within the pod.
- 'persistentVolumeClaim': To use a persistent volume claim for shared data that persists beyond pod restarts.
5. Configure Container Relationships:
- Ensure that the 'name' of the application container and sidecar container is the same as the 'name' used in the 'volumeMounts' section.
Example YAML:



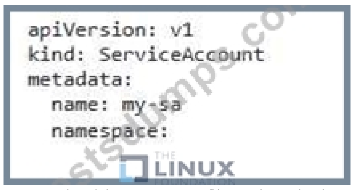
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-app-pod
  labels:
    app: my-app
spec:
  containers:
  - name: app
    image: nginx:latest
    ports:
    - containerPort: 80
  - name: logger
    image: busybox
    command: ["sh", "-c", "while true; do sleep 5; echo 'Logging data'; done"]
    volumeMounts:
    - name: logs
      mountPath: /logs
  volumes:
  - name: logs
    emptyDir: {}
```

- The pod named 'my-app-pod' includes two containers: 'app' (the primary application) and 'logger' (the sidecar). - The 'loggers container uses a 'command' to simulate logging activity. - Both containers can access the 'logs' volume, which is an empty directory.
Important Note: - The sidecar container should ideally be configured to interact with the application container. This might involve using shared volumes, environment variables, or inter-process communication mecnanisms to facilitate data exchange or Signal passing. - Remember to adapt the example to your specific application requirements, choosing the appropriate container images, commands, and volumes.]

NEW QUESTION # 26
You have a Kubernetes cluster with a deployment named 'myapp'. This deployment utilizes a service account named 'my-sas to access a private registry. You need to grant this service account access to pull images from the registry, which requires an image pull secret named 'my-secret How would you configure the service account to use this image pull secret and ensure your myapp' deployment can successfully pull images?

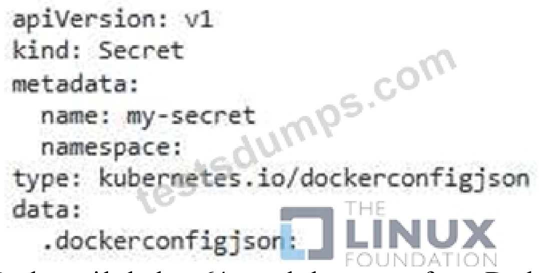
Answer:

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create a Service Account:
- If you haven't already, create a service account named 'my-sa':



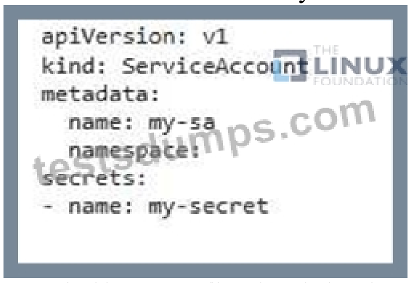
```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-sa
  namespace:
```

- Apply this YAML file using 'kubectl apply -f my-sa.yaml. 2. Create an Image Pull Secret: - Create a secret containing the necessary credentials for your private registry:



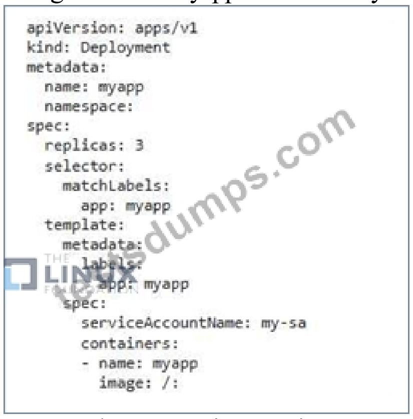
```
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
  namespace:
type: kubernetes.io/dockerconfigjson
data:
  .dockerconfigjson:
```

- Replace with the base64 encoded contents of your Docker configuration file. You can obtain this by using 'cat ~/.docker/config.json | base64'. - Apply the YAML file using 'kubectl apply -f my-secret.yaml' 3. Associate the Secret with the Service Account: - Add the 'my-secret' secret to tne 'my-sa' service account:



```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-sa
  namespace:
secrets:
- name: my-secret
```

- Apply this YAML file using ' kubectl apply -f my-sa_yamr 4. Update Deployment with Service Account - Update the deployment configuration for 'myapp' to use the 'my-sa' service account.



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
  namespace:
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      serviceAccountName: my-sa
      containers:
      - name: myapp
        image: /:
```

- Ensure that 'your-private-registry', 'your-image', and 'your-tag' match the details of your private registry image. - Apply the updated deployment configuration using 'kubectl apply -f myapp.yamr 5. Verify Deployment: - Check the status of the deployment using ' kubectl get deployments myapp'. You should see the pods successfully pulling images from your private registry Important Notes: - Security Best Practices: Always use dedicated service accounts with minimal permissions. - Image Pull Secret: The 'my-secret' secret should be securely stored and managed. - Namespace: Ensure that both the service account and secret are in the same namespace as your deployment. - Registry Authentication: Ensure your private registry is configured with proper authentication for your service account credentials.,

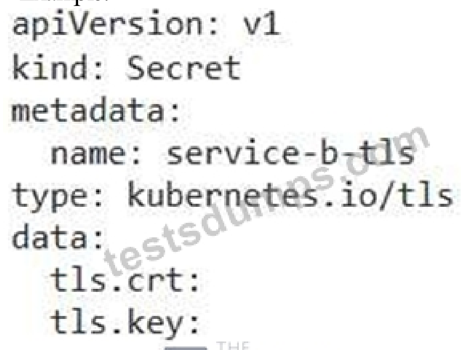
NEW QUESTION # 27

You are building a microservices application on Kubernetes, where two services, and 'service-b' , need to communicate with each other securely. 'Service-b' needs to expose a secure endpoint that is only accessible by 'service-a'. Describe how you would implement this using Kubernetes resources, including the configuration for the 'service-b' endpoint.

Answer:

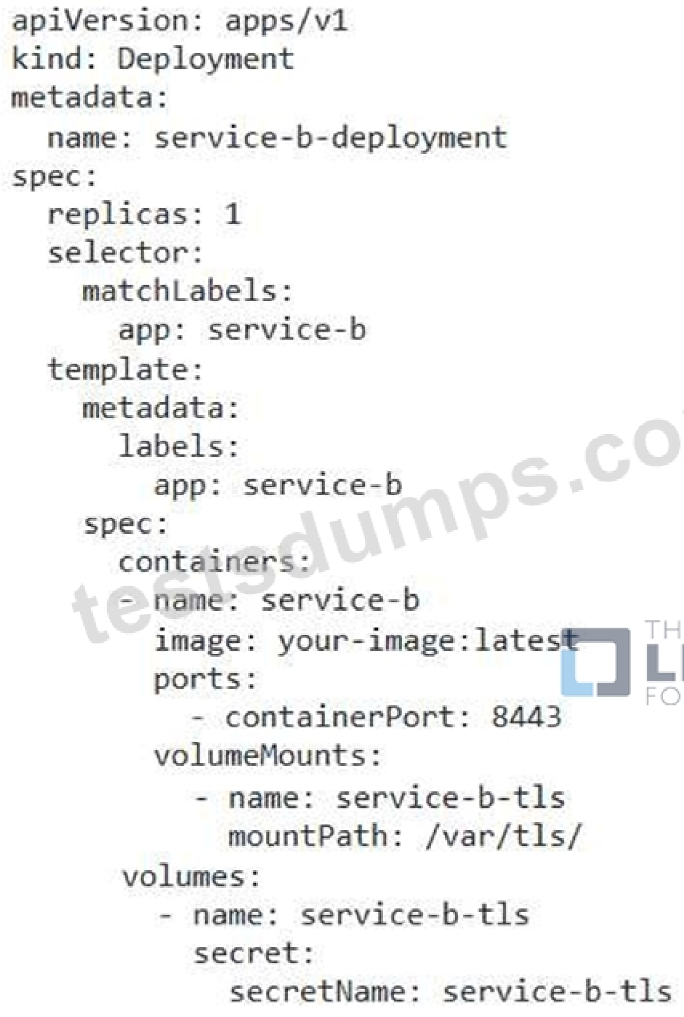
Explanation:
See the solution below with Step by Step Explanation.

Explanation:
Solution (Step by Step) :
1. Define a Kubernetes Secret:
- Create a Kubernetes secret to store the certificate and key pair for 'service-W. This secret will be used to secure the communication.
- Example:



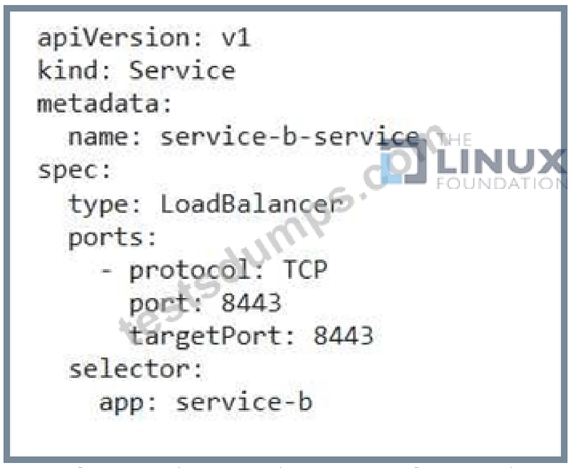
```
apiVersion: v1
kind: Secret
metadata:
  name: service-b-tls
type: kubernetes.io/tls
data:
  tls.crt:
  tls.key:
```

2. Configure 'service-b' Deployment: - Define a Deployment for 'service-b' , specifying a container that uses the secret for TLS. - Ensure that the container has the required dependencies and configuration to use TLS. - Example:



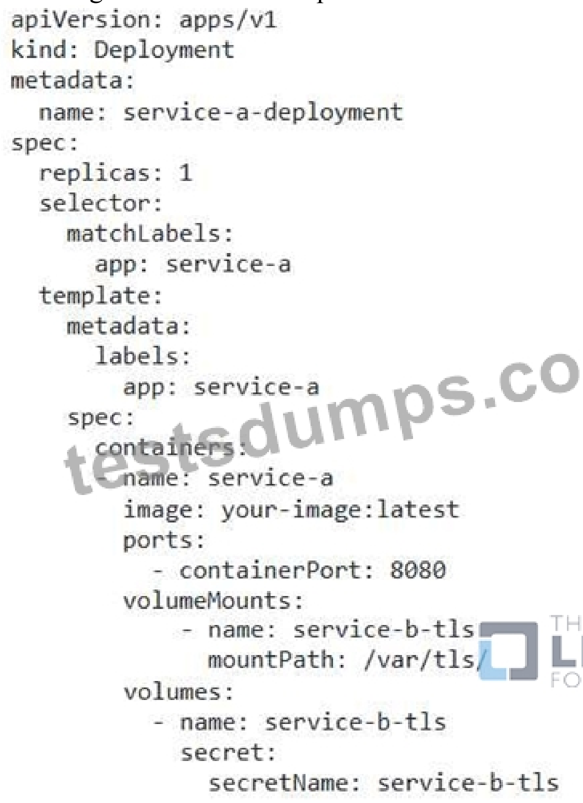
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: service-b-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: service-b
  template:
    metadata:
      labels:
        app: service-b
    spec:
      containers:
      - name: service-b
        image: your-image:latest
        ports:
          - containerPort: 8443
        volumeMounts:
          - name: service-b-tls
            mountPath: /var/tls/
      volumes:
        - name: service-b-tls
          secret:
            secretName: service-b-tls
```

3. Define a Kubernetes Service for 'service-b'.' - Create a Service for 'service-b' that exposes the secure endpoint on a specific port (e.g., 8443) and uses the LoadBalancer' type for external access. - Use the 'targetPort' field to specify the container port that 'service-b' is listening on. - Example:



```yaml
apiVersion: v1
kind: Service
metadata:
  name: service-b-service
spec:
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 8443
      targetPort: 8443
  selector:
    app: service-b
```

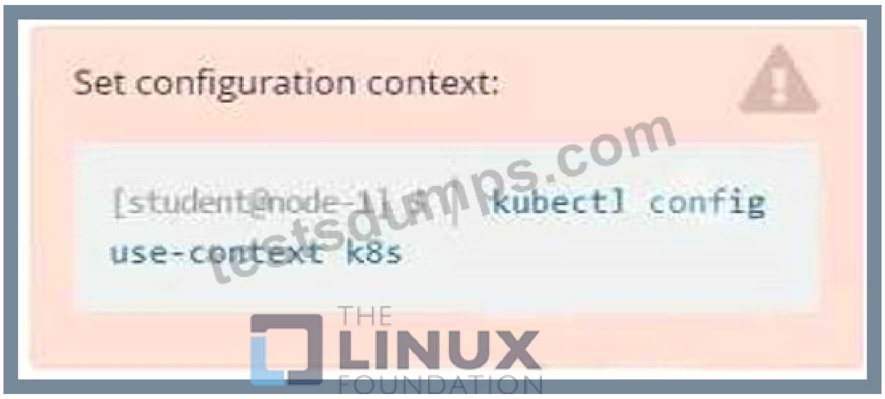
4. Configure 'service-a' Deployment: - Define a Deployment for 'service-a', specifying a container that uses the secret for TLS when connecting to service-W. - Example:



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: service-a-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: service-a
  template:
    metadata:
      labels:
        app: service-a
    spec:
      containers:
      - name: service-a
        image: your-image:latest
        ports:
          - containerPort: 8080
        volumeMounts:
            - name: service-b-tls
              mountPath: /var/tls/
        volumes:
          - name: service-b-tls
            secret:
              secretName: service-b-tls
```

5. Update 'service-a' Container Configuration: - Within the 'service-a' container, ensure the application is configured to use the certificate and key from the mounted volume ('Ivar/tls/') for secure communication with 'service-b'. 6. Verify Secure Communication: - Use 'kubectl get pods' to check the status of both 'service-a' and 'service-W pods. - Test the communication between 'service-a' and 'service-b' by sending requests from the 'service-a' pod to the secure endpoint of 'service-b'. - Verify that the communication is secure and that 'service-a' can successfully access the endpoint. Notes: - You may need to adjust the port numbers and image names in the examples to match your specific setup. - Make sure you have the certificate and key in the correct format and base64 encoded before creating the Secret. - You can also use other methods like a Service Account and Role-Based Access Control (RBAC) to restrict access to the secure endpoint, if needed. - This is a simplified example and additional security measures may be required based on your application's requirements. ,

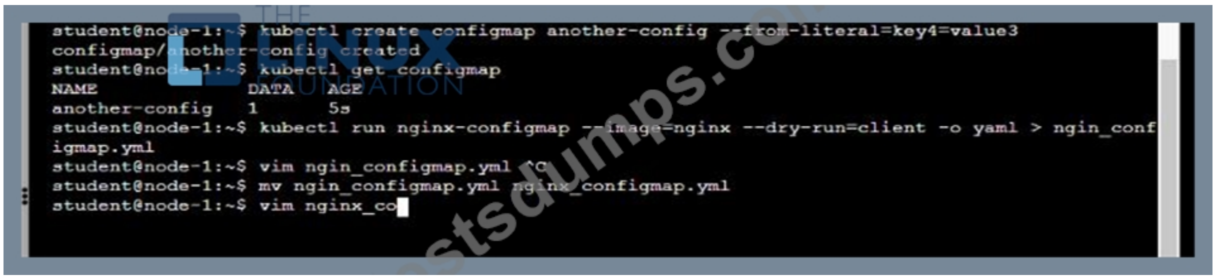
NEW QUESTION # 28
Exhibit:


Set configuration context:

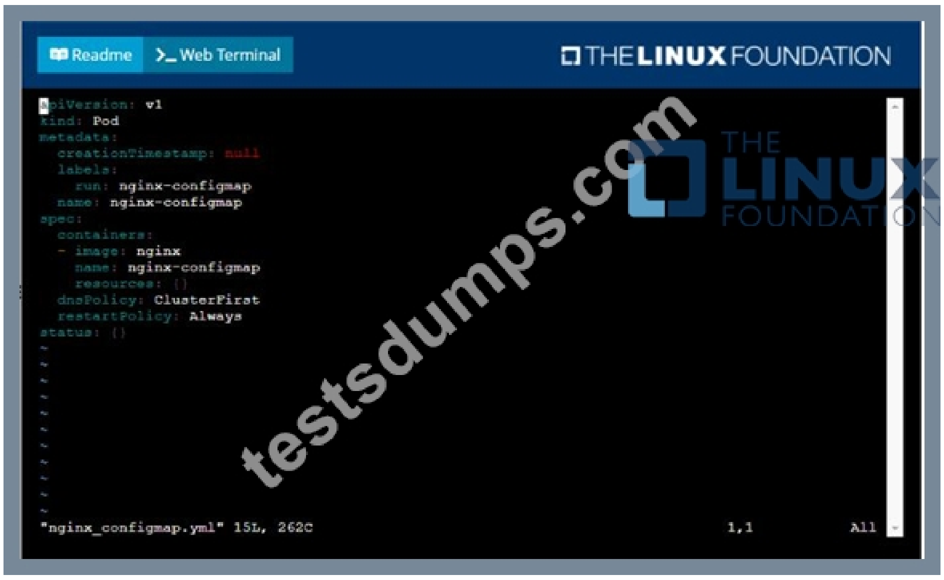
```
[student@node-1] $ | kubectl config
use-context k8s
```

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

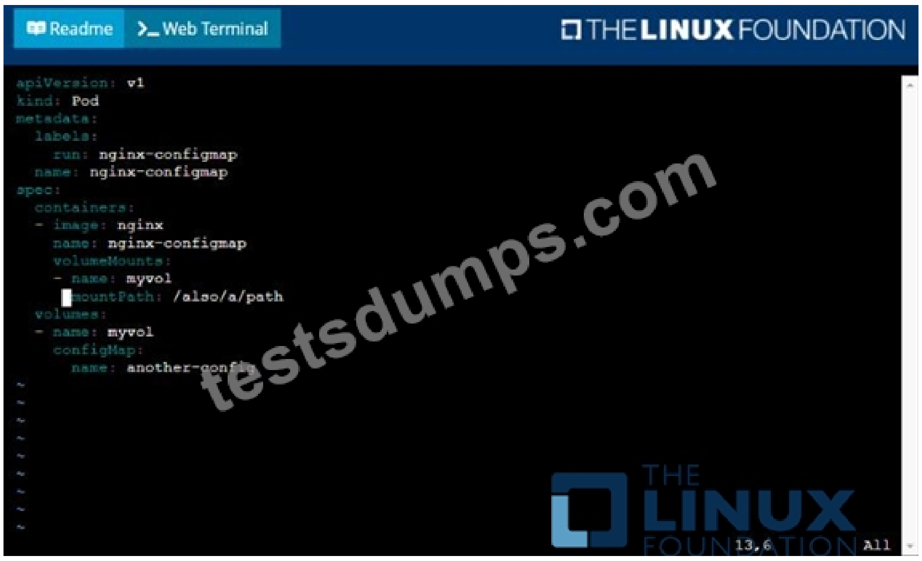
- A. Solution:



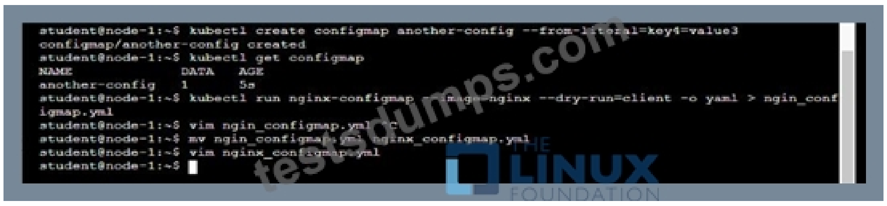
```
student@node-1:~$ kubectl create configmap another-config --from-literal=key4=value3
configmap/another-config created
student@node-1:~$ kubectl get configmap
NAME             DATA   AGE
another-config   1      5s
student@node-1:~$ kubectl run nginx-configmap --image=nginx --dry-run=client -o yaml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



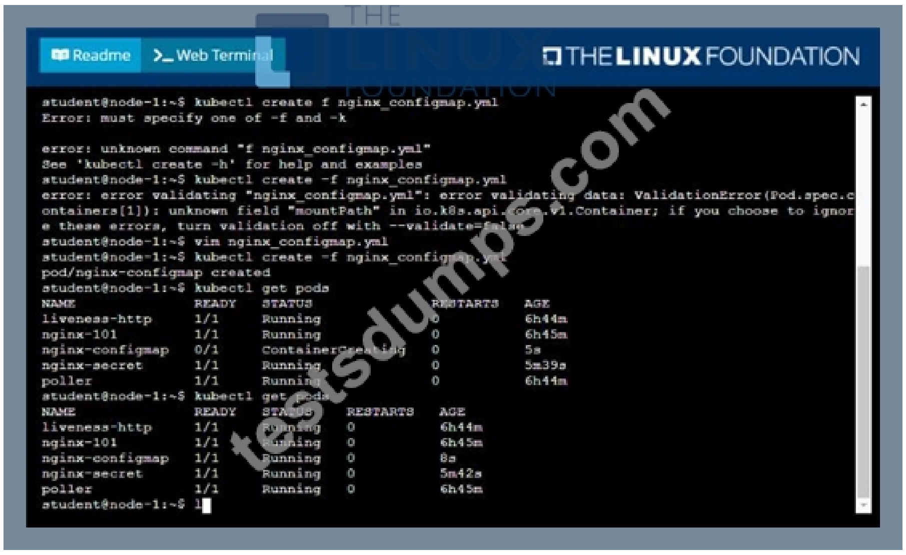
```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx-configmap
  name: nginx-configmap
spec:
  containers:
  - image: nginx
    name: nginx-configmap
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
~
~
~
~
~
~
~
~
~
~
~
"nginx_configmap.yml"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



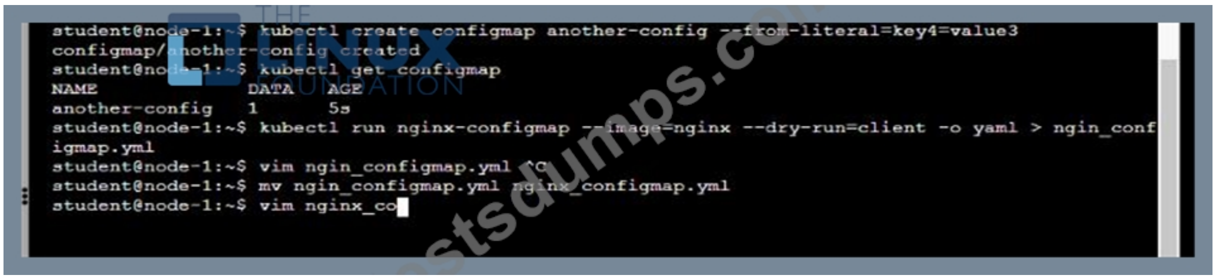
```
student@node-1:~$ kubectl create configmap another-config --from-literal=key4=value3
configmap/another-config created
student@node-1:~$ kubectl get configmap
NAME             DATA   AGE
another-config   1      5s
student@node-1:~$ kubectl run nginx-configmap --image=nginx --dry-run=client -o yaml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



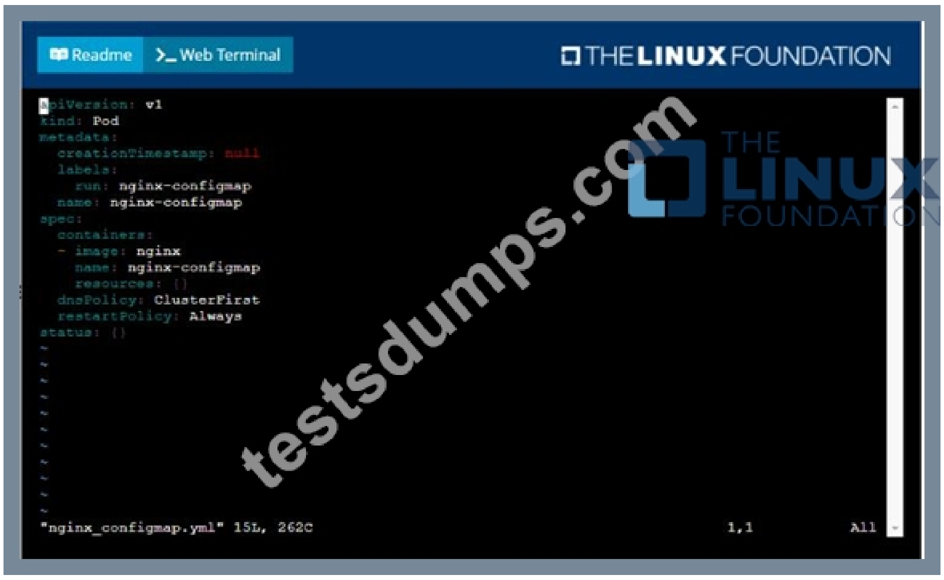
```
student@node-1:~$ kubectl create f nginx_configmap.yml
Error: must specify one of -f and -k

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:~$ 1
```

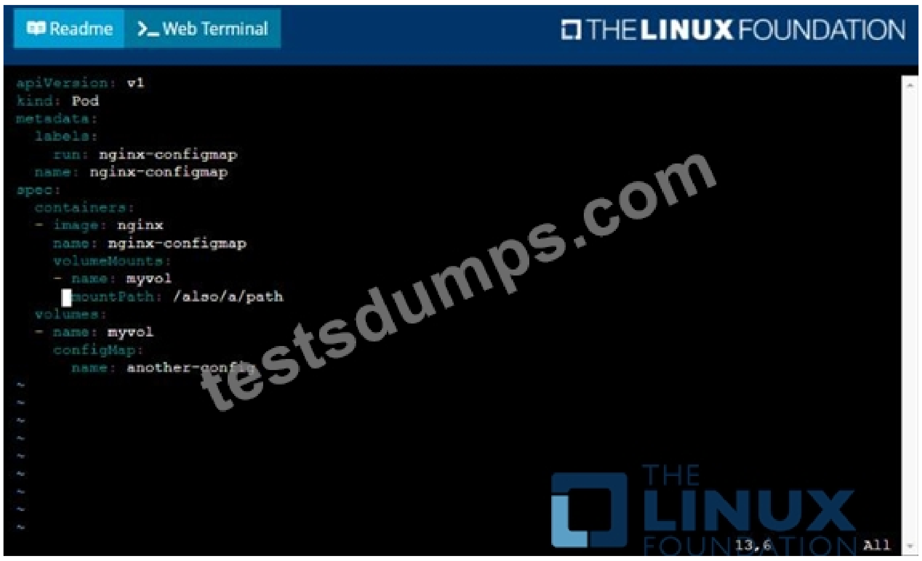
- B. Solution:



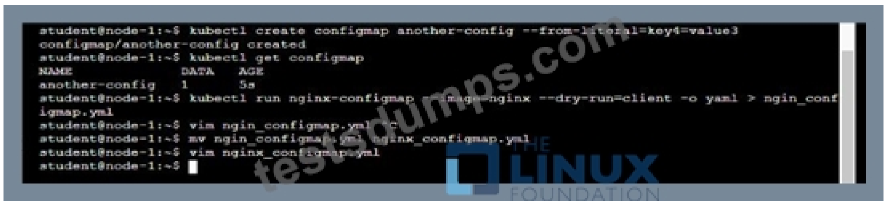
```
student@node-1:~$ kubectl create configmap another-config --from-literal=key4=value3
configmap/another-config created
student@node-1:~$ kubectl get configmap
NAME             DATA   AGE
another-config   1      5s
student@node-1:~$ kubectl run nginx-configmap --image=nginx --dry-run=client -o yaml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



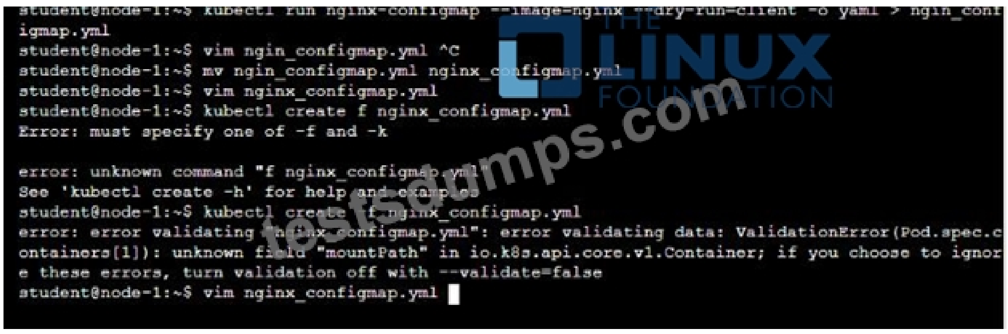
```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx-configmap
  name: nginx-configmap
spec:
  containers:
  - image: nginx
    name: nginx-configmap
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
~
~
~
~
~
~
~
~
~
~
~
"nginx_configmap.yml"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

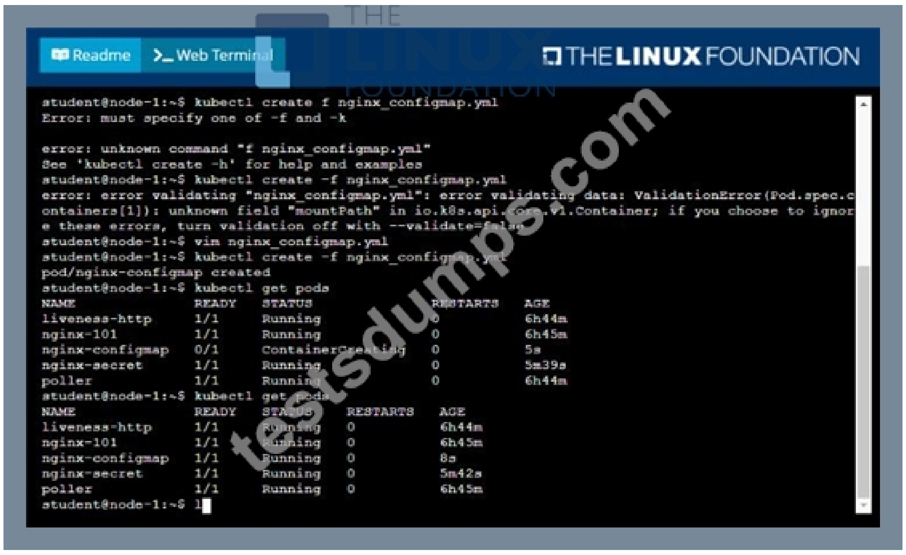


```
student@node-1:~$ kubectl create configmap another-config --from-literal=key4=value3
configmap/another-config created
student@node-1:~$ kubectl get configmap
NAME             DATA   AGE
another-config   1      5s
student@node-1:~$ kubectl run nginx-configmap --image=nginx --dry-run=client -o yaml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



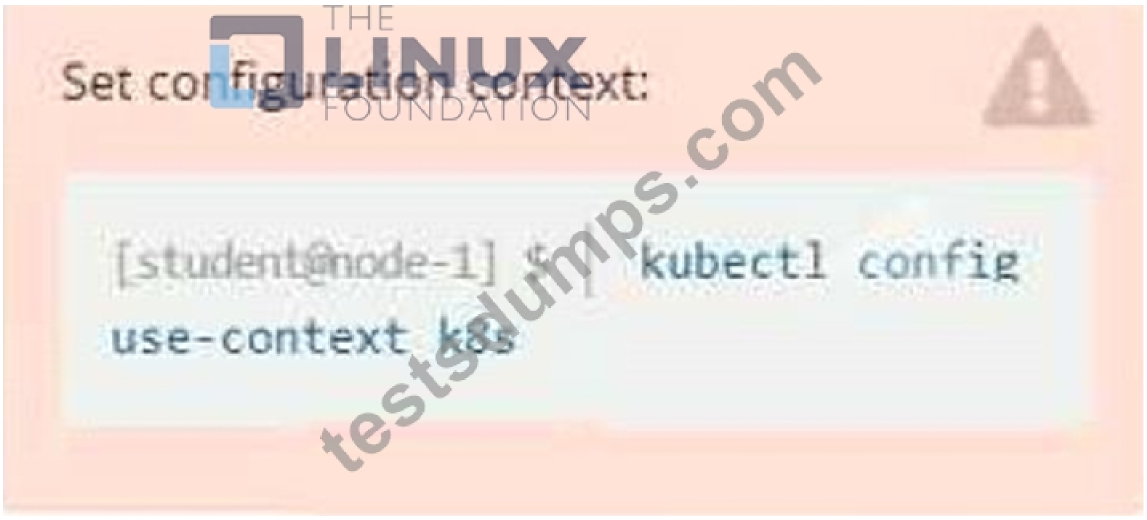
```
student@node-1:~$ kubectl create f nginx_configmap.yml
Error: must specify one of -f and -k

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
```

Answer: B

NEW QUESTION # 29

Exhibit:



Set configuration context:

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

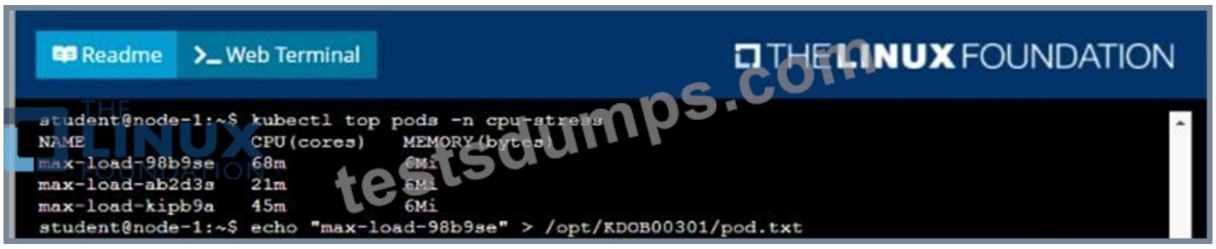
Context

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

Task

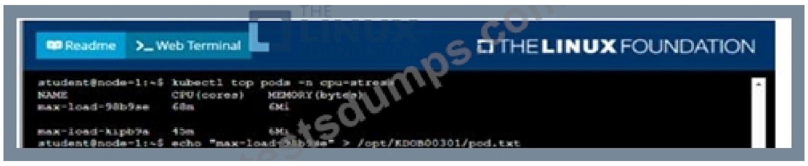
* From the pods running in namespace cpu-stress , write the name only of the pod that is consuming the most CPU to file /opt/KDOBG030l/pod.txt, which has already been created.

- A. Solution:



```
student@node-1:~$ kubectl top pods -n cpu-stress
NAME               CPU(cores)    MEMORY(bytes)
max-load-98b9se    68m           6Mi
max-load-ab2d3s    21m           6Mi
max-load-kipb9a    45m           6Mi
student@node-1:~$ echo "max-load-98b9se" > /opt/KDOB00301/pod.txt
```

- B. Solution:



```
student@node-1:~$ kubectl top pods -n cpu-stress
NAME               CPU(cores)    MEMORY(bytes)
max-load-98b9se    68m           6Mi

max-load-kipb9a    45m           6Mi
student@node-1:~$ echo "max-load-98b9se" > /opt/KDOB00301/pod.txt
```

Answer: A

NEW QUESTION # 30

......

Perhaps your ability cannot meet the requirement of a high salary job. So you cannot get the job because of lack of ability. You must really want to improve yourself. Now, our CKAD exam questions can help you realize your dreams. Not only our CKAD study braindumps can help you obtain the most helpful knowledge and skills to let you stand out by solving the probleme the others can't, but also our CKAD praparation guide can help you get the certification for sure.

CKAD Actualtest: https://www.testsdumps.com/CKAD_real-exam-dumps.html

- Training CKAD Solutions CKAD Exams Dumps Valid CKAD Exam Discount ▸ www.dumpsmaterials.com ◂ is best website to obtain ▹ CKAD ◃ for free download Free CKAD Vce Dumps
- Latest CKAD Exam Question Free CKAD Vce Dumps ☮ Training CKAD Solutions Search for ✔ CKAD ✔ and download exam materials for free through ➡ www.pdfvce.com CKAD Reliable Test Bootcamp
- Reliable CKAD Exam Sample Reliable CKAD Exam Sample Valid CKAD Exam Discount Search for { CKAD } and download it for free on ➤ www.vceengine.com website Training CKAD Solutions
- Test CKAD Questions Pdf CKAD Online Lab Simulation Reliable CKAD Exam Sample The page for free download of 《 CKAD 》 on ✔ www.pdfvce.com ✔ will open immediately CKAD Exams Dumps
- Quiz 2026 Linux Foundation Newest CKAD: Linux Foundation Certified Kubernetes Application Developer Exam Practice Test Fee Search for ⇛ CKAD ⇚ and download it for free on 【 www.examcollectionpass.com 】 website CKAD Valid Test Tutorial
- Quiz 2026 Linux Foundation Newest CKAD: Linux Foundation Certified Kubernetes Application Developer Exam Practice Test Fee Search for ☀ CKAD ☀ and download exam materials for free through ▸ www.pdfvce.com ◂ CKAD Valid Test Tutorial
- Exam CKAD Overviews Premium CKAD Files Valid Dumps CKAD Files Enter { www.pass4test.com } and search for （ CKAD ） to download for free Vce CKAD File
- 100% Pass-Rate CKAD Practice Test Fee Help You to Get Acquainted with Real CKAD Exam Simulation Easily obtain [CKAD] for free download through ➥ www.pdfvce.com New CKAD Exam Simulator
- CKAD - Reliable Linux Foundation Certified Kubernetes Application Developer Exam Practice Test Fee Simply search for ➤ CKAD for free download on www.validtorrent.com CKAD Reliable Test Bootcamp
- CKAD Practice Test Fee - Linux Foundation CKAD Actualtest: Linux Foundation Certified Kubernetes Application Developer Exam Pass for Sure Open 【 www.pdfvce.com 】 enter ➠ CKAD and obtain a free download CKAD Valid Test Tutorial
- 100% Pass-Rate CKAD Practice Test Fee Help You to Get Acquainted with Real CKAD Exam Simulation Open ⇛ www.prep4sures.top ⇚ and search for ☀ CKAD ☀ to download exam materials for free Free CKAD Vce Dumps
- www.stes.tyc.edu.tw, blogfreely.net, qiita.com, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, saviaalquimia.cl, 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, 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, www.stes.tyc.edu.tw, Disposable vapes

P.S. Free & New CKAD dumps are available on Google Drive shared by TestsDumps: https://drive.google.com/open?id=1n_cn2D9yQVh3-qiLjkp3Lq-v3qjdTWPb