Standard CKAD Answers - CKAD Test Braindumps

BONUS!!! Download part of Pass4SureQuiz CKAD dumps for free: https://drive.google.com/open?id=12gaj_NdhvHfTW-6YRXxBn9LH6p4QSfEk

There are numerious CKAD exam dumps for the candidates to select for their preparation the exams, some candidates may get confused by so many choice. Our CKAD learning materials have free demo for the candidates, and they will have a general idea about the CKAD Learning Materials. You can obtain the CKAD learning materials for about ten minutes. The payment is also quite easy: online payment with credit card, and the private information of the you is also guaranteed.

Linux Foundation Certified Kubernetes Application Developer (CKAD) Exam is a certification program offered by the Linux Foundation to validate the skills and knowledge of developers who work with Kubernetes. Kubernetes is an open-source container orchestration system that automates the deployment, scaling, and management of containerized applications. It has become the de facto standard for container orchestration, and the CKAD Certification is recognition of the skills required to deploy and manage applications on Kubernetes.

>> Standard CKAD Answers <<

CKAD Test Braindumps - Latest CKAD Exam Test

Now we can say that Linux Foundation Certified Kubernetes Application Developer Exam (CKAD) exam questions are real and top-notch Linux Foundation CKAD exam questions that you can expect in the upcoming Linux Foundation Certified Kubernetes Application Developer Exam (CKAD) exam. In this way, you can easily pass the CKAD exam with good scores. The countless CKAD Exam candidates have passed their dream CKAD certification exam and they all got help from real, valid, and updated CKAD practice questions, You can also trust on Pass4SureQuiz and start preparation with confidence.

Linux Foundation Certified Kubernetes Application Developer Exam Sample Questions (Q168-Q173):

NEW QUESTION # 168
Context

Task:
A Dockerfile has been prepared at -/human-stork/build/Dockerfile
1) Using the prepared Dockerfile, build a container image with the name macque and lag 3.0. You may install and use the tool of your choice.

2) Using the tool of your choice export the built container image in OC-format and store it at -/human stork/macque 3.0 tar

Answer:

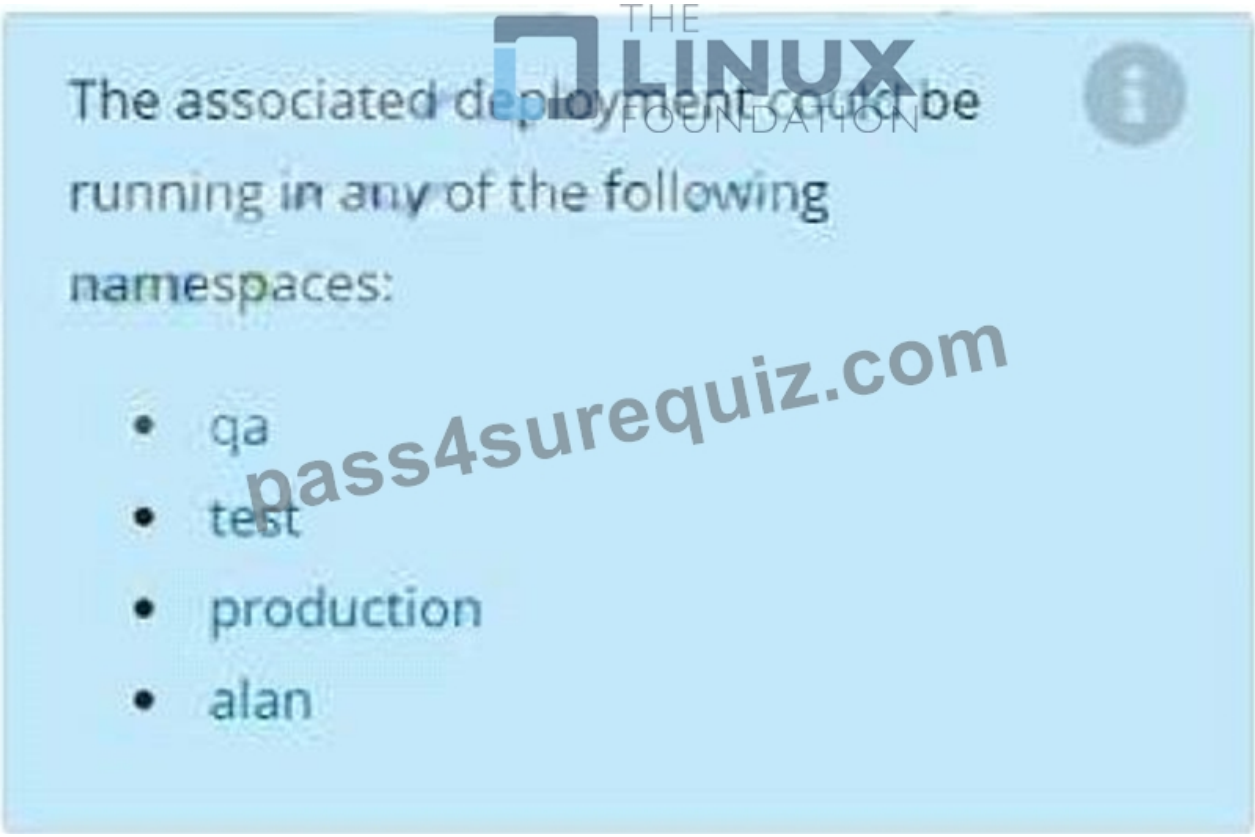
Explanation:
Solution:

```
candidate@node-1:~$ cd humane-stork/build/
candidate@node-1:~/humane-stork/build$ ls -l
total 16
-rw-r--r-- 1 candidate candidate 201 Sep 24 04:21 Dockerfile
-rw-r--r-- 1 candidate candidate 644 Sep 24 04:21 text1.html
-rw-r--r-- 1 candidate candidate 813 Sep 24 04:21 text2.html
-rw-r--r-- 1 candidate candidate 383 Sep 24 04:21 text3.html
candidate@node-1:~/humane-stork/build$ sudo docker build -t macaque:3.0 .
Sending build context to Docker daemon  6.144kB
Step 1/5 : FROM docker.io/lfccncf/nginx:mainline
 ---> ea335eea17ab
Step 2/5 : ADD text1.html /usr/share/nginx/html/
 ---> 8967ee9ee5d0
Step 3/5 : ADD text2.html /usr/share/nginx/html/
 ---> cb0554422f26
Step 4/5 : ADD text3.html /usr/share/nginx/html/
 ---> 62e879ab821e
Step 5/5 : COPY text2.html /usr/share/nginx/html/index.html
 ---> 331c8a94372c
Successfully built 331c8a94372c
Successfully tagged macaque:3.0
candidate@node-1:~/humane-stork/build$ sudo docker save macaque:3.0 > ~/humane-stork/macaque-3.0.tar
candidate@node-1:~/humane-stork/build$ cd ..
candidate@node-1:~/humane-stork$ ls -l
total 142532
drwxr-xr-x 2 candidate candidate      4096 Sep 24 04:21 build
-rw-rw-r-- 1 candidate candidate 145948672 Sep 24 11:39 macaque-3.0.tar
candidate@node-1:~/humane-stork$
```

NEW QUESTION # 169
Exhibit:

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:

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.

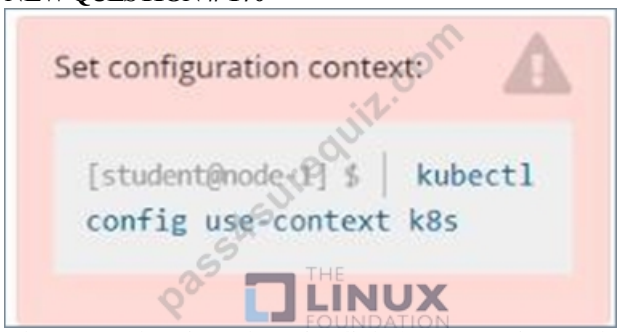


- A. Solution:
  Create the Pod:
  kubectl create -f http://k8s.io/docs/tasks/configure-pod-container/exec-liveness.yaml
  Within 30 seconds, view the Pod events:
  kubectl describe pod liveness-exec
  The output indicates that no liveness probes have failed yet:
  FirstSeen LastSeen Count From SubobjectPath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

The output indicates that no liveness probes have failed yet:
FirstSeen LastSeen Count From SubobjectPath Type Reason Message
--------- -------- ----- ---- ------------- -------- ------ -------
24s 24s 1 {default-scheduler } Normal Scheduled Successfully assigned liveness-exec to worker0
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image "gcr.io/google_containers/busybox"
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image "gcr.io/google_containers/busybox"
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Created Created container with docker id 86849c15382e; Security:[seccomp=unconfined]
23s 23s 1 {kubelet worker0} spec.containers{liveness} Normal Started Started container with docker id 86849c15382e
After 35 seconds, view the Pod events again:
kubectl describe pod liveness-exec
At the bottom of the output, there are messages indicating that the liveness probes have failed, and the containers have been killed and recreated.
FirstSeen LastSeen Count From SubobjectPath Type Reason Message
--------- -------- ----- ---- ------------- -------- ------ -------
37s 37s 1 {default-scheduler } Normal Scheduled Successfully assigned liveness-exec to worker0
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulling pulling image "gcr.io/google_containers/busybox"
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Pulled Successfully pulled image "gcr.io/google_containers/busybox"
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Created Created container with docker id 86849c15382e; Security:[seccomp=unconfined]
36s 36s 1 {kubelet worker0} spec.containers{liveness} Normal Started Started container with docker id 86849c15382e
2s 2s 1 {kubelet worker0} spec.containers{liveness} Warning Unhealthy Liveness probe failed: cat: can't open '/tmp/healthy': No such file or directory
Wait another 30 seconds, and verify that the Container has been restarted:
kubectl get pod liveness-exec
The output shows that RESTARTS has been incremented:
NAME READY STATUS RESTARTS AGE
liveness-exec 1/1 Running 1 m

Answer: B

NEW QUESTION # 170



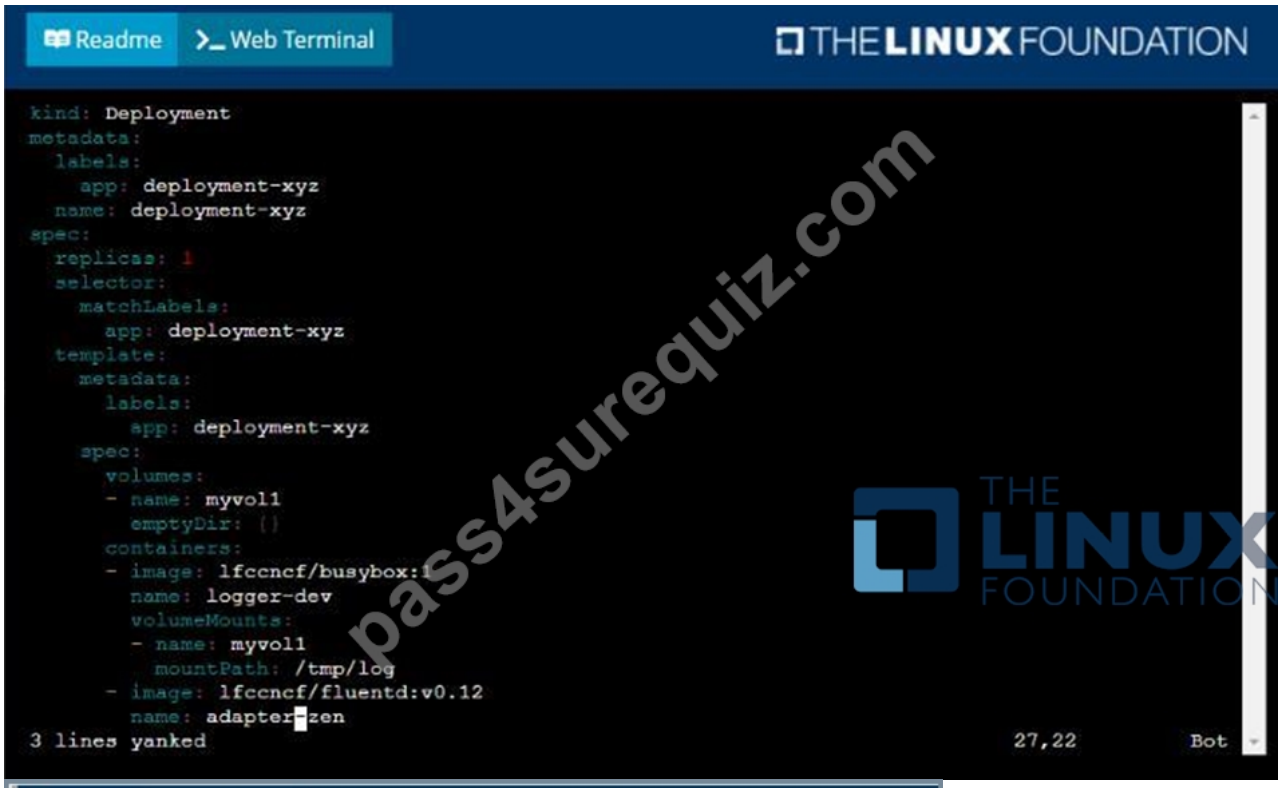
Given a container that writes a log file in format A and a container that converts log files from format A to format B, create a deployment that runs both containers such that the log files from the first container are converted by the second container, emitting logs in format B.
Task:
* Create a deployment named deployment-xyz in the default namespace, that:
*Includes a primary
lfccncf/busybox:1 container, named logger-dev
*includes a sidecar lfccncf/fluentd:v0.12 container, named adapter-zen
*Mounts a shared volume /tmp/log on both containers, which does not persist when the pod is deleted
*Instructs the logger-dev
container to run the command

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

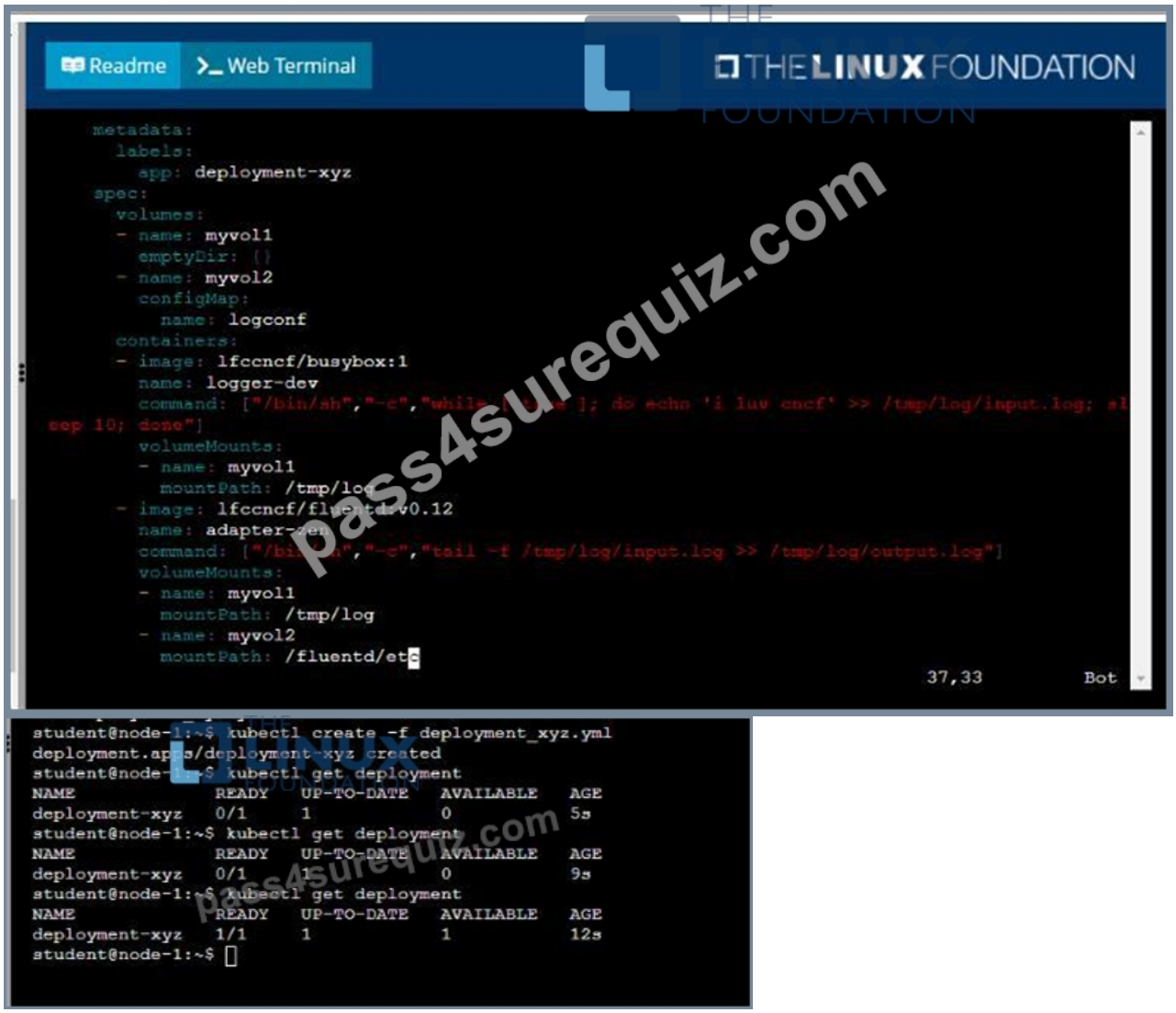
* The adapter-zen sidecar container should read /tmp/log/input.log and output the data to /tmp/log/output.* in Fluentd JSON format. Note that no knowledge of Fluentd is required to complete this task: all you will need to achieve this is to create the ConfigMap from the spec file provided at /opt/KDMC00102/fluentd-configma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

Answer:

Explanation:
Explanation
Solution:


Readme
Web Terminal
THE LINUX FOUNDATION
kind: Deployment
metadata:
labels:
app: deployment-xyz
name: deployment-xyz
spec:
replicas: 1
selector:
matchLabels:
app: deployment-xyz
template:
metadata:
labels:
app: deployment-xyz
spec:
volumes:
- name: myvol1
emptyDir: {}
containers:
- image: lfccncf/busybox:1
name: logger-dev
volumeMounts:
- name: myvol1
mountPath: /tmp/log
- image: lfccncf/fluentd:v0.12
name: adapter-zen
3 lines yanked
27,22
Bot
pass4surequiz.com
THE LINUX FOUNDATION


Readme
Web Terminal
THE LINUX FOUNDATION
replicas: 1
selector:
matchLabels:
app: deployment-xyz
template:
metadata:
labels:
app: deployment-xyz
spec:
volumes:
- name: myvol1
emptyDir: {}
containers:
- image: lfccncf/busybox:1
name: logger-dev
command: ["/bin/sh","-c","while true ; do echo 'i luv cncf' >> /tmp/log/input.log; sl
eep 10; done"]
volumeMounts:
- name: myvol1
mountPath: /tmp/log
- image: lfccncf/fluentd:v0.12
name: adapter-zen
command: ["/bin/sh","-c","tail -f /tmp/log/input.log >> /tmp/log/output.log"]
volumeMounts:
- name: myvol1
mountPath: /tmp/log
29,83
Bot
pass4surequiz.com
THE LINUX FOUNDATION



NEW QUESTION # 171
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:

```
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:

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 # 172
You have a Kubernetes application that uses a custom resource definition (CRD) to manage its configuration. The application logs are written to a dedicated container log file. You want to use Kustomize to automate the process of fetching and displaying these logs. How can you achieve this using Kustomize and a custom resource?

Answer:

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

1. Define the Custom Resource:
- Create a custom resource definition (CRD) that defines the structure of

NEW QUESTION # 173
......

It is not hard to know that CKAD torrent prep is compiled by hundreds of industry experts based on the syllabus and development trends of industries that contain all the key points that may be involved in the examination. Therefore, with CKAD exam questions, you no longer need to purchase any other review materials, and you also don't need to spend a lot of money on tutoring classes. At the same time, CKAD Test Guide will provide you with very flexible learning time in order to help you pass the exam.