# CKAD최신버전덤프공부 - CKAD높은통과율덤프공부


EC-COUNCIL 312-39 Certified SOC Analyst (CSA) 4

.....

312-39퍼펙트 덤프 최신버전: https://www.passtip.net/312-39-pass-exam.html

- 시험대비 312-39덤프공부 덤프공부문제 □ □ www.itdumpskr.com □의 무료 다운로드[ 312-39 ]페이지가 지금 열립니다312-39최신버전 시험덤프문제
- 높은 통과율 312-39덤프공부 시험대비자료 □ 무료 다운로드를 위해□ 312-39 □□□를 검색하려면➡ www.itdumpskr.com ⬅을(를) 입력하십시오312-39덤프문제모음
- 312-39시험대비 최신버전 덤프 □ 312-39합격보장 가능 인증덤프 □ 312-39인증시험대비자료 □ 시험 자료를 무료로 다운로드하려면□ www.itdumpskr.com □을 통해□ 312-39 □를 검색하십시오312-39높은 통과율 덤프공부
- 312-39인증시험 덤프문제 ◀ 312-39시험패스 가능한 공부자료 □ 312-39인증시험 덤프문제 □▶ www.itdumpskr.com ◀에서 검색만 하면□ 312-39 □□□를 무료로 다운로드할 수 있습니다312-39시험대비 최신버전 덤프
- 312-39퍼펙트 최신버전 덤프자료 □ 312-39합격보장 가능 인증덤프 □ 312-39합격보장 가능 공부 □▷ www.itdumpskr.com ◁에서 검색만 하면□ 312-39 □를 무료로 다운로드할 수 있습니다312-39합격보장 가능 인증덤프
- 312-39인증시험대비자료 □ 312-39퍼펙트 공부문제 □ 312-39퍼펙트 최신버전 덤프자료 □□ www.itdumpskr.com □에서□ 312-39 □를 검색하고 무료 다운로드 받기312-39자격증덤프
- 312-39덤프공부 최신 덤프샘플문제 □ { www.itdumpskr.com }웹사이트에서□ 312-39 □를 열고 검색하여 무료 다운로드312-39최고패스자료
- 312-39덤프공부 최신 덤프샘플문제 □ □ www.itdumpskr.com □웹사이트를 열고□ 312-39 □를 검색하여 무료 다운로드312-39최신버전 시험대비 공부자료
- 312-39덤프공부최신버전 인증덤프 □ 시험 자료를 무료로 다운로드하려면□ www.itdumpskr.com □을 통해□ 312-39 □를 검색하십시오312-39높은 통과율 덤프공부
- 퍼펙트한 312-39덤프공부 덤프 최신버전 □ □ www.itdumpskr.com □웹사이트를 열고[ 312-39 ]를 검색하여 무료 다운로드312-39최신버전 시험대비 공부자료
- 312-39덤프공부 최신 덤프로 시험패스 도전! □ □ www.itdumpskr.com □에서 검색만 하면▶ 312-39 ◀를 무료로 다운로드할 수 있습니다312-39적중율 높은 인증덤프공부

Tags: 312-39덤프공부,312-39퍼펙트 덤프 최신버전,312-39인기자격증 덤프공부자료,312-39인증덤프샘플다운,312-39시험덤프공부

312-39덤프공부 - 312-39퍼펙트 덤프최신버전

ITDumpsKR CKAD 최신 PDF 버전 시험 문제집을 무료로 Google Drive에서 다운로드하세요: https://drive.google.com/open?id=1l6Ee7nmiDvqhMbqiFSuPja7eS6Q3gGaL

ITDumpsKR는 IT인증자격증을 취득하려는 IT업계 인사들의 검증으로 크나큰 인지도를 가지게 되었습니다. 믿고 애용해주신 분들께 감사의 인사를 드립니다. Linux Foundation CKAD덤프도 다른 과목 덤프자료처럼 적중율 좋고 통과율이 장난이 아닙니다. 덤프를 구매하시면 퍼펙트한 구매후 서비스까지 제공해드려 고객님이 보유한 덤프가 항상 시장에서 가장 최신버전임을 약속해드립니다. Linux Foundation CKAD덤프만 구매하신다면 자격증 취득이 쉬워져 고객님의 밝은 미래를 예약한것과 같습니다.

CKAD 시험은 Kubernetes 환경에서 일련의 코딩 도전 과제를 완료하는 것을 요구하는 실습 중심 시험입니다. 시험은 온라인으로 진행되며 감독관이 있어 시험의 진정성을 보장합니다. 시험 시간은 2시간이며 이 동안 Kubernetes 객체를 생성하고 수정하고, 네트워킹 및 보안을 구성하며 Kubernetes 환경에서 발생하는 일반적인 문제를 해결할 수 있는 능력을 증명해야 합니다.

CKAD 인증 시험을 준비하려면 개인은 Kubernetes 응용 프로그램 개발에 대한 실습 경험이 있어야합니다. 또한 Linux Foundation은 인증 시험에 통과하는 데 필요한 지식과 기술을 개인에게 제공하는 CKAD 교육 과정을 제공합니다. 시험을 준비하는 데 사용할 수있는 다양한 온라인 리소스 및 실습 시험도 있습니다.

**>> CKAD최신버전 덤프공부 <<**

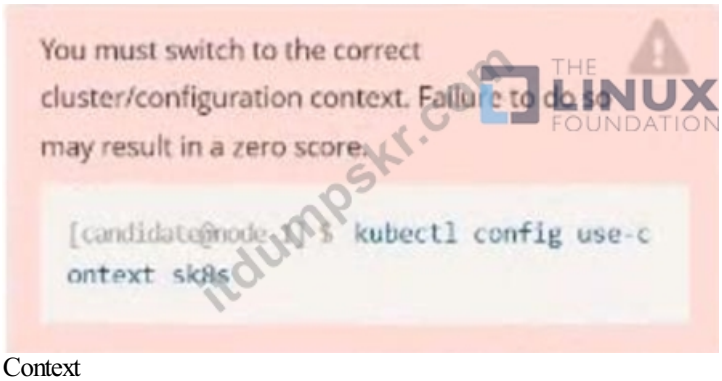
## CKAD높은 통과율 덤프공부 & CKAD응시자료

ITDumpsKR는 여러분의 요구를 만족시켜드리는 사이트입니다. 많은 분들이 우리사이트의 it인증덤프를 사용함으로 관련it시험을 안전하게 패스를 하였습니다. 이니 우리 ITDumpsKR사이트의 단골이 되었죠. ITDumpsKR에서는 최신의Linux Foundation CKAD자료를 제공하며 여러분의Linux Foundation CKAD인증시험에 많은 도움이 될 것입니다.

CKAD 자격증은 컨테이너 기반 애플리케이션 개발 경험이 있고 Kubernetes 애플리케이션 개발 능력을 검증하고자 하는 소프트웨어 개발자를 대상으로 합니다. 또한 Kubernetes와 작업하는 시스템 관리자, DevOps 엔지니어, IT 전문가도 자격증을 취득하여 기술과 지식을 향상시킬 수 있습니다.

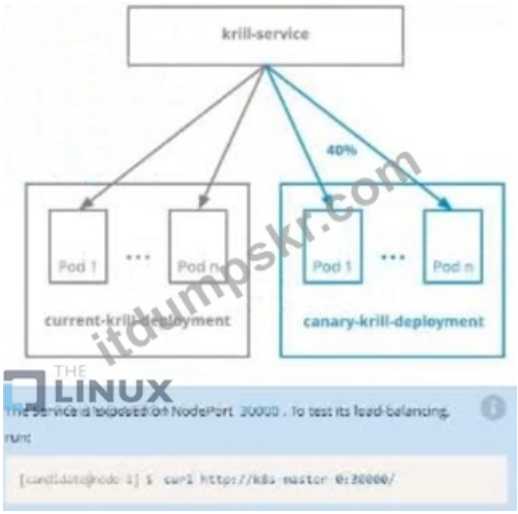
# 최신 Kubernetes Application Developer CKAD 무료샘플문제 (Q20-Q25):

**질문 # 20**
Refer to Exhibit.



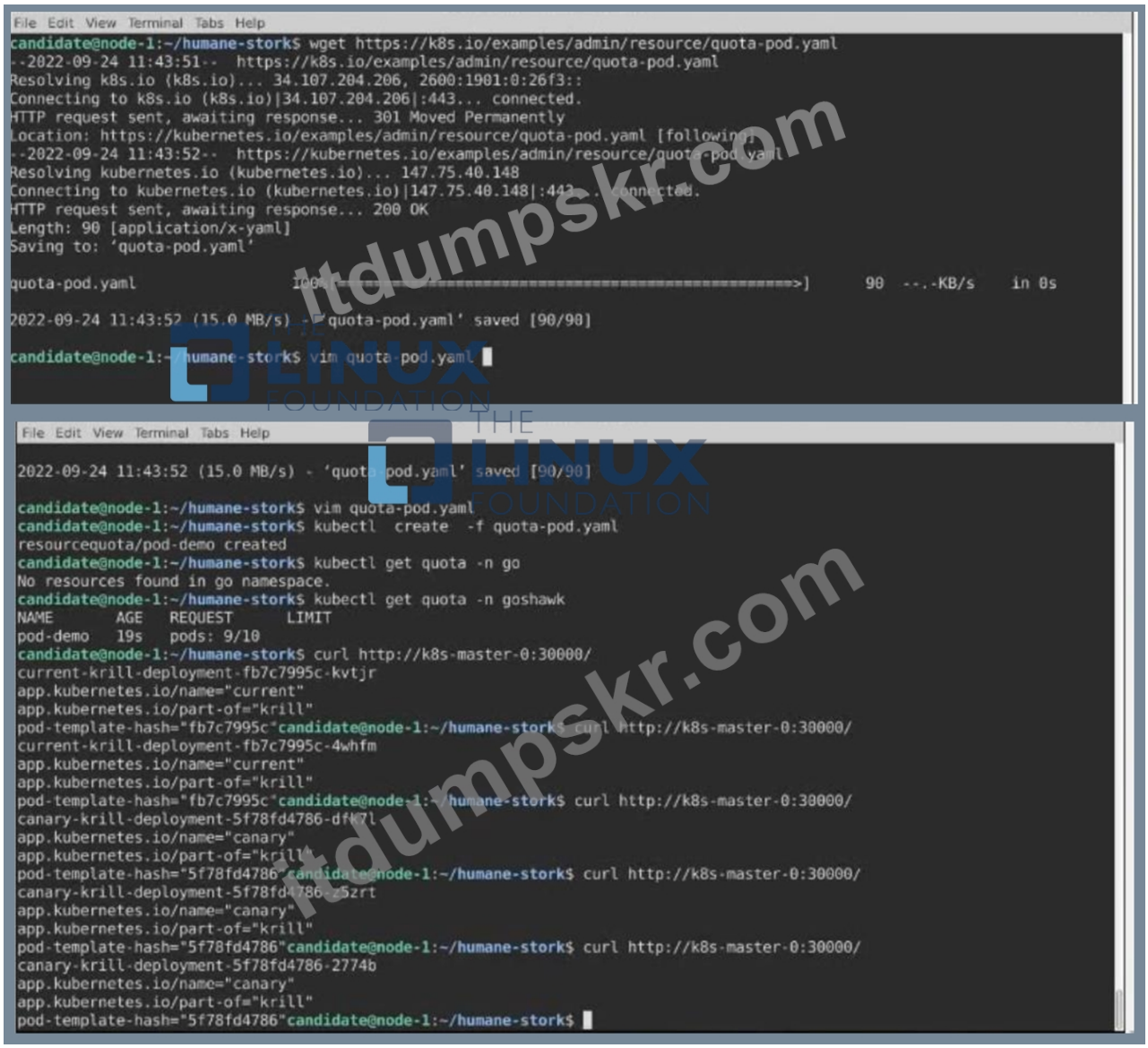

Context
You are asked to prepare a Canary deployment for testing a new application release.
Task:
A Service named krill-Service in the goshark namespace points to 5 pod created by the Deployment named current-krill-deployment

1) Create an identical Deployment named canary-kill-deployment, in the same namespace.
2) Modify the Deployment so that:
-A maximum number of 10 pods run in the goshawk namespace.
-40% of the krill-service 's traffic goes to the canary-krill-deployment pod(s)




**정답：**

**설명：**
Solution:




**질문 # 21**
You nave a multi-container pod tnat uses a database container and an application container. The database container is responsible for storing sensitive datm You need to ensure that the database container only runs on nodes that have a specific label, like 'sensitive-data=true', for added security. How would you implement this constraint using Pod Affinity and node selectors?

**정답：**

**설명：**
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Define Node Labels:
- First, you need to label your nodes appropriately. I-Jse ' kubectl label nodes sensitive-data=true' for nodes that should host the database container
2. Create Pod Affinity Rules:
- In your pod definition, add a 'nodeselector' to enforce the requirement.

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
  - name: database
    image: example/database:latest
    ports:
    - containerPort: 5432
  - name: app
    image: example/app:latest
    ports:
    - containerPort: 8080
  nodeSelector:
    sensitive-data: "true"
```

3. Apply the Pod Definition: - Apply the updated pod definition using 'kubectl apply -f my-app-yaml' 4. Verification: - Check the pod status using 'kubectl get pods my-app'. Ensure the pod is scheduled on a node with the 'sensitive-data-true label. 5. Further Security: - You can additionally use 'podAffinity' to ensure that the database container and the application container run on different nodes. This adds an extra layer of security in case one node is compromised.

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
  - name: database
    image: example/database:latest
    ports:
    - containerPort: 5432
  - name: app
    image: example/app:latest
    ports:
    - containerPort: 8080
  nodeSelector:
    sensitive-data: "true"
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: "app"
            operator: In
            values:
            - "database"
        topologyKey: "kubernetes.io/hostname"
```

- In this example, 'podAntiAtfinitys Witn 'requiredDuringScnedulinglgnoredDunngExecution' ensures tnat tne database container and tne application container are not scheduled on the same node, thus preventing potential data breaches. ,

**질문 # 22**


Task:
1) Fix any API depreciation issues in the manifest file -/credible-mite/www.yaml so that this application can be deployed on cluster K8s.

2) Deploy the application specified in the updated manifest file -/credible-mite/www.yaml in namespace cobra See the solution below.

**정답：**

**설명：**

Explanation

Solution:

```
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ vim ~/credible-mite/www.yaml
```

Text Description automatically generated

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: www-deployment
  namespace: cobra
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: "nginx:stable"
          ports:
            - containerPort: 80
          volumeMounts:
            - mountPath: /var/log/nginx
              name: logs
          env:
            - name: NGINX_ENTRYPOINT_QUIET_LOGS
              value: "1"
      volumes:
        - name: logs
          emptyDir: {}
```

Text Description automatically generated

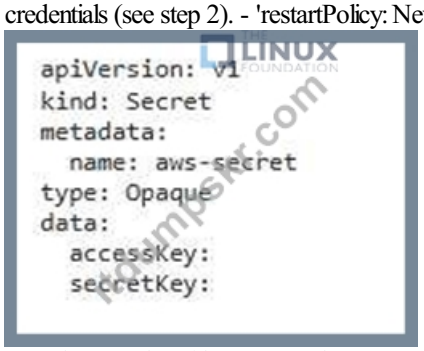
```
File  Edit  View  Terminal  Tabs  Help
deployment.apps/expose created
candidate@node-1:~$ kubectl get pods -n ckad00014
NAME                      READY   STATUS              RESTARTS   AGE
expose-85dd99d4d9-25675   0/1     ContainerCreating   0          6s
expose-85dd99d4d9-4fhcc   0/1     ContainerCreating   0          6s
expose-85dd99d4d9-fld7j   0/1     ContainerCreating   0          6s
expose-85dd99d4d9-tt6rm   0/1     ContainerCreating   0          6s
expose-85dd99d4d9-vjd8b   0/1     ContainerCreating   0          6s
expose-85dd99d4d9-vtzpq   0/1     ContainerCreating   0          6s
candidate@node-1:~$ kubectl  get deploy -n ckad00014
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
expose   6/6     6            6           15s
candidate@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
candidate@node-1:~$ vim ~/credible-mite/www.yaml
candidate@node-1:~$ vim ~/credible-mite/www.yaml
candidate@node-1:~$ kubectl  apply  -f ~/credible-mite/www.yaml
deployment.apps/www-deployment created
candidate@node-1:~$ kubectl get pods -n cobra
NAME                              READY   STATUS              RESTARTS   AGE
www-deployment-d899c6b49-d6ccg    1/1     Running             0          6s
www-deployment-d899c6b49-f796l    0/1     ContainerCreating   0          6s
www-deployment-d899c6b49-ztfcw    0/1     ContainerCreating   0          6s
candidate@node-1:~$ kubectl get deploy -n cobra
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
www-deployment   3/3     3            3           11s
candidate@node-1:~$ kubectl get pods -n cobra
NAME                              READY   STATUS    RESTARTS   AGE
www-deployment-d899c6b49-d6ccg    1/1     Running   0          14s
www-deployment-d899c6b49-f796l    1/1     Running   0          14s
www-deployment-d899c6b49-ztfcw    1/1     Running   0          14s
candidate@node-1:~$
```

**질문 # 23**
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?

**정답：**

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


- 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:




- 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.

**질문 # 24**
You are building a web application with two microservices: a frontend service ('frontend') and a backend service ( ' backend'). The frontend service requires access to the backend service, which iS exposed on port 8080 within the Kubernetes cluster. How would you configure an Ingress resource to direct traffic to the correct service based on the hostname, ensuring that the frontend service can access the backend service internally without exposing the backend service to the public internet?

**정답：**

**설명：**
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create a Service tor the Backend Service:
- Define a Service for the 'backend' service, exposing it internally within the Kubernetes cluster on port 8080.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  ports:
    - port: 8080
      targetPort: 8080
  selector:
    app: backend
```



2. Configure the Ingress Resource: - Create an Ingress resource that directs traffic to the frontend service based on the hostname, allowing the frontend service to access the backend service internally without exposing it to the public internet - Define the Ingress

rule to map the hostname 'frontend-example.com' to the 'frontend' service on port 80. - Configure an Ingress rule to enable access to the 'backend' service on port 8080 using the hostname 'internal-backend-example-com' within the Kubernetes cluster.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
  - host: frontend.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: frontend-service
            port:
              number: 80
  - host: internal-backend.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: backend-service
            port:
              number: 8080
  tls:
  - hosts:
    - frontend.example.com
    secretName: frontend-tls
```

3. Create a Secret for the Frontend TLS Certificate: - Create a Secret in Kubernetes to store the TLS certificate and key for the frontend service.

```
apiVersion: v1
kind: Secret
metadata:
  name: frontend-tls
type: tls
data:
  tls.crt:
  tls.key:
```

4. Apply the Resources: - Apply the Service, Ingress, and Secret YAML files to your Kubernetes cluster using 'kubectl apply -f 5. Access the Frontend Service: - Access the frontend service using the hostname 'frontend-example.com'. The frontend service can now access the backend service internally using the hostname 'internal-backend-example-com' without exposing the backend service to the public internet.]

**질문 # 25**

.....

- 시험패스에 유효한 CKAD최신버전 덤프공부 덤프샘플문제 ♣ 무료로 다운로드하려면▷ www.itdumpskr.com ◁로 이동하여➥ CKAD □를 검색하십시오CKAD 100% 시험패스 덤프
- CKAD최신버전 덤프공부 시험준비에 가장 좋은 인기덤프 □ ➤ kr.fast2test.com □에서《 CKAD 》를 검색하고 무료 다운로드 받기CKAD인증덤프 샘플문제
- wiishlist.com, laraibrc983805.blog5star.com, alvinnibi874994.verybigblog.com, jaysonkcmp001558.wikiinside.com, bookmarkstumble.com, saulktis256665.luwebs.com, businessbookmark.com, orangebookmarks.com, miriamfzgn945784.blog5star.com, inesvvvr074569.wikiconverse.com, Disposable vapes

그 외, ITDumpsKR CKAD 시험 문제집 일부가 지금은 무료입니다: https://drive.google.com/open?id=1l6Ee7nmiDvqhMbqiFSuPja7eS6Q3gGaL