Linux Foundation Exam CKAD Study Guide: Linux Foundation Certified Kubernetes Application Developer Exam - ITExamDownload 10 Years of Excellence

What's more, part of that ITExamDownload CKAD dumps now are free: https://drive.google.com/open?id=1d_5CHCsriy9uEeuXJDoWI8eLkOZktEzZ

ITExamDownload designed this prep material to help you pass the exam on the first try. It may sound complicated, but once you go through regular study and intensive practice, passing the final exam would be a piece of cake. The cost of Linux Foundation Certified Kubernetes Application Developer Exam (CKAD) certification itself is expensive, ranging from $100 to $1000, so you can't risk wasting that amount. ITExamDownload ensures that this does not happen by providing you with reliable and updated preparation material.

Linux Foundation Certified Kubernetes Application Developer (CKAD) certification exam is designed to test a candidate's knowledge and skills in developing cloud-native applications on Kubernetes. Kubernetes is an open-source container orchestration system that has become the de facto standard for managing containerized applications in production environments. As the demand for Kubernetes skills increases, the CKAD Certification is becoming a popular choice for developers who want to demonstrate their expertise in Kubernetes application development.

>> Exam CKAD Study Guide <<

CKAD Preparation | CKAD Latest Test Labs

Linux Foundation CKAD dumps may be the best method for candidates who are preparing for their exam and eager to clear exam as soon as possible. People's success lies in their good use of every change to self-improve. Our Linux Foundation CKAD Dumps will be the best resources for your real test. If you choose our products, we will choose efficient and high-passing preparation materials.

Linux Foundation CKAD (Certified Kubernetes Application Developer) Certification Exam is a popular certification exam for individuals who want to showcase their proficiency in Kubernetes application development. Kubernetes is an open-source system used for automating deployment, scaling, and management of containerized applications. As Kubernetes gains popularity, the demand for professionals with CKAD Certification is rapidly increasing. Linux Foundation Certified Kubernetes Application Developer Exam certification exam is designed to test the candidate's ability to design, build, configure, and expose Kubernetes applications.

Linux Foundation Certified Kubernetes Application Developer Exam Sample Questions (Q151-Q156):

NEW QUESTION # 151

Context
A web application requires a specific version of redis to be used as a cache.
Task
Create a pod with the following characteristics, and leave it running when complete:
* The pod must run in the web namespace.
The namespace has already been created
* The name of the pod should be cache
* Use the Ifccncf/redis image with the 3.2 tag
* Expose port 6379

Answer:

Explanation:
See the solution below.
Explanation
Solution:

NEW QUESTION # 152
You have a Deployment named 'nginx-deployment' running 3 replicas of an Nginx container. You need to ensure that all 3 pods are using the same ConfigMap for configuration. Additionally, you need to configure the ConfigMap so that changes made to it are automatically reflected in the running pods without requiring a new Deployment update.

Answer:

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

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-config
data:
  nginx_conf: |
    worker_processes  1;
    events {
      worker_connections  1024;
    }
    http {
      server {
        listen 80;
        location / {
          root /usr/share/nginx/html;
          index index.html index.htm;
        }
      }
    }
```

2. Apply the ConfigMap: bash kubectl apply -f nginx-config_yaml 3. Update the Deployment to use the ConfigMap:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        volumeMounts:
        - name: nginx-config-volume
          mountPath: /etc/nginx/conf.d
      volumes:
      - name: nginx-config-volume
        configMap:
          name: nginx-config
```

4. Apply the updated Deployment bash kubectl apply -f nginx-deployment.yaml 5. Verify the Deployment: bash kubectl get deployments nginx-deployment You should see that the Deployment is using the 'nginx-config' ConfigMap for its configuration. 6. Test the automatic update: - Modify the 'nginx-config' ConfigMap: bash kubectl edit configmap nginx-config Change the 'nginx_conf' value in the ConfigMap. - Verify the change in the pods: bash kubectl exec -it -- bash -c 'cat /etc/nginxfconf-d/nginx_conf' Replace with the name of one of the pods- This command will display the contents of the nginx configuration file within the pod. You will observe that the nginx configuration file in the running pods is automatically updated without needing a Deployment update.

NEW QUESTION # 153
You have a Deployment named 'wordpress-deployment' that runs a WordPress application. You want to ensure that Kubernetes automatically restarts pods if tney experience an unexpected termination, such as a container crasn. Implement the necessary configuration for your deployment.

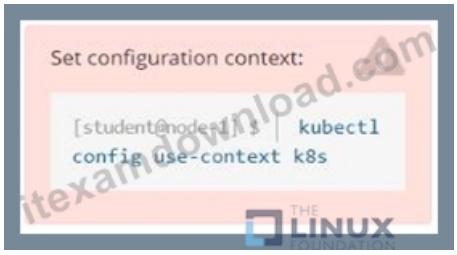
Answer:

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
I). Update the Deployment YAML:
- Add the 'restartpolicy: Always to the 'spec.template_spec.containers' section of your Deployment YAML. This ensures that the pod will always be restarted if a container terminates unexpectedly.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: wordpress
  template:
    metadata:
      labels:
        app: wordpress
    spec:
      containers:
      - name: wordpress
        image: wordpress:latest
        restartPolicy: Always
```

2. Apply the Deployment - Apply the updated Deployment YAML using: bash kubectl apply -f wordpress-deployment-yaml 3. Test the Restart Policy: - Simulate a container crash within a pod (e.g., by sending a SIGKILL Signal to the container). - Observe the pod status using 'kuactl get pods -l app=wordpress' . You snould see the pod being automatically restarted, and the 'STATUS should become 'Running' again. Important Note: - The restaAPolicy: Always' is the default setting for Kubernetes deployments. By explicitly adding it to your YAML, you ensure that this behavior is documented and consistent within your deployment configuration.,

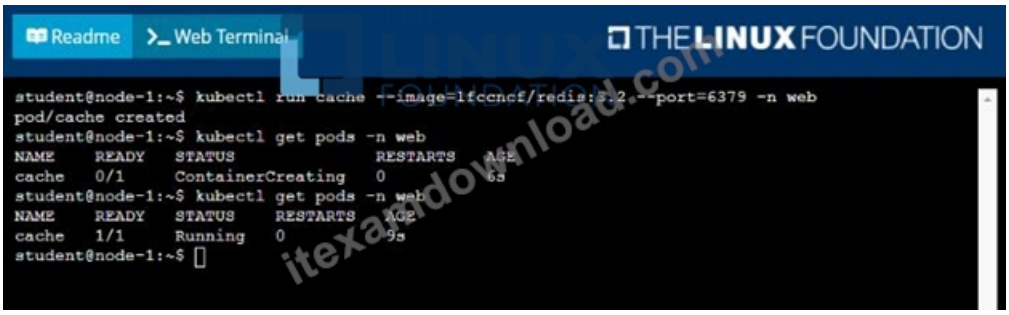
NEW QUESTION # 154
Exhibit:



Context
A web application requires a specific version of redis to be used as a cache.
Task
Create a pod with the following characteristics, and leave it running when complete:
* The pod must run in the web namespace.
The namespace has already been created
* The name of the pod should be cache
* Use the Ifccncf/redis image with the 3.2 tag
* Expose port 6379

- A. Solution:

- B. Solution:



Answer: B

NEW QUESTION # 155
Context

Context
A container within the poller pod is hard-coded to connect the nginxsvc service on port 90 . As this port changes to 5050 an additional container needs to be added to the poller pod which adapts the container to connect to this new port. This should be realized as an ambassador container within the pod.
Task
* Update the nginxsvc service to serve on port 5050.
* Add an HAproxy container named haproxy bound to port 90 to the poller pod and deploy the enhanced pod. Use the image haproxy and inject the configuration located at /opt/KDMC00101/haproxy.cfg, with a ConfigMap named haproxy-config, mounted into the container so that haproxy.cfg is available at /usr/local/etc/haproxy/haproxy.cfg. Ensure that you update the args of the poller container to connect to localhost instead of nginxsvc so that the connection is correctly proxied to the new service endpoint. You must not modify the port of the endpoint in poller's args . The spec file used to create the initial poller pod is available in /opt/KDMC00101/poller.yaml

Answer:

Explanation:
Solution:
apiVersion: apps/v1
kind: Deployment
metadata:
name: my-nginx
spec:
selector:
matchLabels:
run: my-nginx
replicas: 2
template:
metadata:
labels:
run: my-nginx

spec:
containers:
- name: my-nginx
image: nginx
ports:
- containerPort: 90
This makes it accessible from any node in your cluster. Check the nodes the Pod is running on:
kubectl apply -f ./run-my-nginx.yaml
kubectl get pods -l run=my-nginx -o wide
NAME READY STATUS RESTARTS AGE IP NODE
my-nginx-3800858182-jr4a2 1/1 Running 0 13s 10.244.3.4 kubernetes-minion-905m my-nginx-3800858182-kna2y 1/1 Running 0 13s 10.244.2.5 kubernetes-minion-ljyd Check your pods' IPs:
kubectl get pods -l run=my-nginx -o yaml | grep podIP
podIP: 10.244.3.4
podIP: 10.244.2.5

NEW QUESTION # 156
......