# CKAD dumps materials - exam dumps for CKAD: Linux Foundation Certified Kubernetes Application Developer Exam


BTW, DOWNLOAD part of BootcampPDF CKAD dumps from Cloud Storage: https://drive.google.com/open?id=1iXqEaVDvmhdizv7Dw0I0aAzoh1UWyJ6X

BootcampPDF CKAD desktop and web-based practice exams are distinguished by their excellent features. The CKAD web-based practice exam is supported by all operating systems and can be taken through popular browsers including Chrome, MS Edge, Internet Explorer, Opera, Firefox, and Safari. Windows computers can run the desktop Linux Foundation CKAD Practice Test software. You won't require a live internet connection to use the desktop Linux Foundation exam simulation software once you've verified the product's license.

Just look at the text version of the introduction, you may still be unable to determine whether this product is suitable for you, or whether it is worth your purchase. We are very fond of preparing trial versions of our CKAD study materials for you so that you can have a clearly check on not only the content of the CKAD Exam Braindumps, but also the displays. The content of the tiral version is a small part of our CKAD practice questions, and it is easy and convenient to free download.

**>> Advanced CKAD Testing Engine <<**

## CKAD Latest Test Preparation | Latest CKAD Exam Test

BootcampPDF is responsible for our CKAD study materials. Every exam product of BootcampPDF have sold to customer will enjoy considerate after-sales service. If you have problems about our CKAD study materials such as installation, operation and so on, we will quickly reply to you after our online workers have received your emails. We are not afraid of troubles. We warmly welcome to your questions and suggestions on the CKAD Exam Questions. We sincerely hope we can help you solve your problem and help you pass the CKAD exam.

The CKAD certification exam is an ideal certification for developers who are looking to advance their careers in the field of cloud-native application development. Linux Foundation Certified Kubernetes Application Developer Exam certification exam is designed to help developers demonstrate their skills and knowledge of Kubernetes and to stand out from the crowd in a competitive job market. Linux Foundation Certified Kubernetes Application Developer Exam certification also helps organizations to identify qualified professionals who can help them to build and manage Kubernetes applications.

Linux Foundation CKAD Exam is an ideal certification for IT professionals who want to enhance their skills in cloud-native application development and deployment. Linux Foundation Certified Kubernetes Application Developer Exam certification validates a candidate's knowledge of Kubernetes resources, application design and development, debugging, troubleshooting, and security. To prepare for the exam, candidates can take advantage of various resources provided by the Linux Foundation and join

the Kubernetes community to gain insights into best practices.

Linux Foundation CKAD Exam is designed to help developers demonstrate their expertise in deploying and managing applications on Kubernetes. Linux Foundation Certified Kubernetes Application Developer Exam certification is recognized by the industry as a benchmark for Kubernetes skills, making it a valuable asset for developers who want to advance their careers.

## Linux Foundation Certified Kubernetes Application Developer Exam Sample Questions (Q32-Q37):

**NEW QUESTION # 32**
You have a Deployment for a web application that uses a separate Redis cache pod for session management. You want to ensure that each web application pod can access a dedicated Redis instance for session management and avoid contention. Explain how you can use a PersistentVolumeClaim and StatetulSet to achieve this.

**Answer:**

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
I). Create a PersistentVoIumeClaim:
- Create a PersistentVoIumeCIaim (PVC) to request persistent storage for each Redis instance.
- Specify the storage size, access modes, and other requirements based on your needs.
- Example:
2. Create a StatefulSet: - Define a StatefulSet for the Redis pods. - Associate each Redis pod with a unique PVC, ensuring that each pod gets its own dedicated persistent volume. - Example:
3. Configure the Deployment: - Denne tne web application pod Within a Deployment. - Use a Service (e.g., 'redis-service') to access the Redis instances. - Make sure the web application's code can access the Redis instances using the service name. - Example:
4. Deploy and Test - Deploy tne PersistentVolumeClaim, StatefulSet, and Deployment. - Test the web application to ensure that each pod can access its own dedicated Redis instance and session data is correctly managed Without contention. 5. Important: - StatefulSets ensure that each pod has a unique hostname and persistent storage, making them suitable for managing stateful applications. - This approach helps isolate Redis instances, preventing session data conflicts and ensuring the scalability of your web application. ,

**NEW QUESTION # 33**
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:
See the solution below.
Explanation:
Solution:

**NEW QUESTION # 34**
You are deploying a new application named 'analytics-platform' that requires 12 replicas. You want to implement a rolling update strategy that allows for a maximum ot three pods to be unavailable at any given time. However, you need to ensure that the update process iS triggered automatically whenever a new image is pushed to the Docker Hub repository 'analytics/analytics-platform:latest'.

**Answer:**

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Llpdate the Deployment YAMLI
- Update the 'replicas' to 12.
- Define 'maxiJnavailable: 3' and 'maxSurge: in the 'strategy-rollingupdate' sectiom
- Configure a 'strategy.type' to 'RollinglJPdates to trigger a rolling update wnen the deployment is updated.
- Add a 'spec-template.spec.imagePullPolicy: Always' to ensure that the new image is pulled even if it exists in the pod's local cache.
2. Create the Deployment: - Apply the updated YAML file using 'kubectl apply -f analytics-platform-deployment.yamr 3. Verify the Deployment - Check the status of the deployment using 'kubectl get deployments analytics-platform-deployment' to confirm the rollout and updated replica count. 4. Trigger the Automatic Update: - Push a new image to the 'analytics/analytics-platform:latest Docker Hub repository 5. Monitor the Deployment - Use 'kubectl get pods -l app=analytics-platform' to monitor the pod updates during the rolling update process. You will observe that three pods are terminated at a time, while three new pods with the updated image are created. 6. Check for Successful Update: - Once the deployment is complete, use 'kubectl describe deployment analytics-platform-deployment to see that the 'updatedReplicas' field matches the 'replicas' field, indicating a successful update.,

**NEW QUESTION # 35**
Context
You are asked to set resource requests and limits for a running workload to ensure fair resource management.
"Do not delete the existing Deployment . Failure to do so will result in a reduced score." Next, ensure that the total amount of resources in the namespace matches the maximum resources the Pods from the nginx-resources Deployment can request.
Failure to do so will result in the updated Deployment failing to roll out successfully.

**Answer:**

Explanation:
See the Explanation below for complete solution.
Explanation:
Below are the exact steps/commands you can run.
1) Locate the Deployment and its namespace
kubectl get deploy -A | grep nginx-resources
You should see output like:
<namespace> nginx-resources ...
Set a variable (replace <NS> with what you see):
NS=<NS>
Confirm replicas:
kubectl -n $NS get deploy nginx-resources -o jsonpath='{.spec.replicas}{"\n"}'
2) Check if there is a ResourceQuota in that namespace
kubectl -n $NS get resourcequota
kubectl -n $NS describe resourcequota
If there is a quota, note these fields (common ones):
* requests.cpu
* requests.memory
* limits.cpu
* limits.memory
3) Decide requests/limits for the Deployment (example values)
If the question (in your environment) provides specific values, use those.
If it doesn't, a typical safe pair is:
* requests: cpu: 100m, memory: 128Mi
* limits: cpu: 200m, memory: 256Mi
I'll proceed with these example values. If your lab specifies different numbers, just swap them in.
4) Update the existing Deployment (DO NOT DELETE)
Option A (fastest): kubectl set resources
Assuming the container name is the first container (we'll detect it):
kubectl -n $NS get deploy nginx-resources -o jsonpath='{.spec.template.spec.containers[*].name}{"\n"}' If it prints a single container name, set it like this:
kubectl -n $NS set resources deploy nginx-resources \

--requests=cpu=100m,memory=128Mi \
--limits=cpu=200m,memory=256Mi
Verify the Deployment now has resources
kubectl -n $NS get deploy nginx-resources -o jsonpath='{.spec.template.spec.containers[0].resources}{"\n"}'
5) Compute the total resources requested by the Deployment
Get replicas:
REPLICAS=$(kubectl -n $NS get deploy nginx-resources -o jsonpath='{.spec.replicas}') echo $REPLICAS
6) Ensure the namespace quota matches (or exceeds) those totals
This is the part the question warns about: if the quota is too low, the Deployment update will fail to roll out.
6.1 If a ResourceQuota already exists
Patch it to allow at least the totals you calculated.
First, identify quota name:
RQ=$(kubectl -n $NS get resourcequota -o jsonpath='{.items[0].metadata.name}') echo $RQ Then patch (example: replicas=2 # requests.cpu=200m, requests.memory=256Mi):
kubectl -n $NS patch resourcequota $RQ --type='merge' -p '{
"spec": {
"hard": {
"requests.cpu": "200m",
"requests.memory": "256Mi",
"limits.cpu": "400m",
"limits.memory": "512Mi"
}
}
}'
Adjust those numbers to your replicas × request, and replicas × limit (if your quota also enforces limits).
6.2 If there is NO ResourceQuota
Create one that matches the Deployment max request totals.
Example for replicas=2 with our sample requests/limits:
cat <<EOF | kubectl apply -n $NS -f -
apiVersion: v1
kind: ResourceQuota
metadata:
name: nginx-resources-quota
spec:
hard:
requests.cpu: "200m"
requests.memory: "256Mi"
limits.cpu: "400m"
limits.memory: "512Mi"
EOF
7) Verify rollout succeeds
kubectl -n $NS rollout status deploy nginx-resources
kubectl -n $NS get pods
Verify the running pods actually have the requests/limits:
kubectl -n $NS get pod -l app=nginx-resources -o jsonpath='{range .items[*]}{.metadata.name}{" "}{.spec.containers[0].resources}{"\n"}{end}'
(If the label selector app=nginx-resources doesn't exist, just pick a pod name from kubectl get pods and run:) kubectl -n $NS describe pod <pod-name> | sed -n '/Limits:/,/Requests:/p' Common reasons this fails (and the fix)
* Rollout stuck / pods pending with "exceeded quota"Check:
* kubectl -n $NS describe pod <pending-pod>
* kubectl -n $NS describe resourcequota
Fix: increase ResourceQuota hard values to match required totals.
* You set requests higher than quota allowsFix: either reduce requests or raise quota.
kubectl get deploy -A | grep nginx-resources
kubectl -n <NS> get deploy nginx-resources -o jsonpath='{.spec.replicas}{"\n"}{.spec.template.spec.containers[0].name}{"\n"}'
kubectl -n <NS> describe resourcequota

**NEW QUESTION # 36**

You have a Deployment for a web application that uses a separate Redis cache pod for session management. You want to ensure that each web application pod can access a dedicated Redis instance for session management and avoid contention. Explain how you can use a PersistentVolumeClaim and StatetulSet to achieve this.

**Answer:**

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
I). Create a PersistentVoIumeClaim:
- Create a PersistentVoIumeCIaim (PVC) to request persistent storage for each Redis instance.
- Specify the storage size, access modes, and other requirements based on your needs.
- Example:
2. Create a StatefulSet: - Define a StatefulSet for the Redis pods. - Associate each Redis pod with a unique PVC, ensuring that each pod gets its own dedicated persistent volume. - Example:
3. Configure the Deployment: - Denne tne web application pod Within a Deployment. - Use a Service (e.g., 'redis-service') to access the Redis instances. - Make sure the web application's code can access the Redis instances using the service name. - Example:
4. Deploy and Test - Deploy tne PersistentVolumeClaim, StatefulSet, and Deployment. - Test the web application to ensure that each pod can access its own dedicated Redis instance and session data is correctly managed Without contention. 5. Important: - StatefulSets ensure that each pod has a unique hostname and persistent storage, making them suitable for managing stateful applications. - This approach helps isolate Redis instances, preventing session data conflicts and ensuring the scalability of your web application. ,

**NEW QUESTION # 37**
.....

- kelas.fauzan.icu, www.stes.tyc.edu.tw, www.notebook.ai, hhi.instructure.com, dorahacks.io, letterboxd.com, www.myvrgame.cn, www.stes.tyc.edu.tw, dl.instructure.com, www.stes.tyc.edu.tw, Disposable vapes

P.S. Free 2026 Linux Foundation CKAD dumps are available on Google Drive shared by BootcampPDF: https://drive.google.com/open?id=1iXqEaVDvmhdizv7Dw0I0aAzoh1UWyJ6X