Linux Foundation CKAD인기자격증인증시험자료, CKAD최신버전인기덤프

그 외, DumpTOP CKAD 시험 문제집 일부가 지금은 무료입니다: https://drive.google.com/open?id=1nN62W2lqqZd17D0mmGZYfbvwYcxGuUh1

DumpTOP의 Linux Foundation CKAD덤프는 IT업계에 오랜 시간동안 종사한 전문가들의 끊임없는 노력과 지금까지의 노하우로 만들어낸Linux Foundation CKAD시험대비 알맞춤 자료입니다. DumpTOP의 Linux Foundation CKAD덤프만 공부하시면 여러분은 충분히 안전하게 Linux Foundation CKAD시험을 패스하실 수 있습니다. DumpTOP Linux Foundation CKAD덤프의 도움으로 여러분은 IT업계에서 또 한층 업그레이드 될것입니다

Linux Foundation CKAD 시험은 Kubernetes에 애플리케이션을 배포하고 관리하는 전문성을 개발자가 증명하는 데 도움이 되도록 설계되었습니다. 이 자격증은 Kubernetes 기술에 대한 기준으로 인식되어 개발자들이 경력을 발전시키고자 하는 경우 가치 있는 자산입니다.

>> Linux Foundation CKAD인기자격증 인증시험자료 <<

CKAD최신버전 인기덤프 - CKAD최신버전 시험대비자료

DumpTOP는Linux Foundation CKAD인증시험의 촉매제 같은 사이트입니다.Linux Foundation CKAD인증시험 관연 덤프가 우리DumpTOP에서 출시되었습니다. 여러분이Linux Foundation CKAD인증시험으로 나 자신과 자기만의 뛰어난 지식 면을 증명하고 싶으시다면 우리 DumpTOP의Linux Foundation CKAD덤프자료가 많은 도움이 될 것입니다.

CKAD 시험은 명령 줄 도구를 사용하여 Kubernetes 응용 프로그램 개발 및 배포에서 개발자의 숙련도를 테스트하도록 설계되었습니다. 시험은 후보자가 라이브 Kubernetes 클러스터 환경에서 작업을 수행 해야하는 19 가지 질문으로 구성됩니다. 시험은 시간이 지남에 따라, 후보자들은이를 완료하기 위해 2 시간이 주어집니다. 인증 프로그램은 공급 업체-중립적이므로 특정 클라우드 제공 업체와 관련이 없으며 전 세계적으로 인정됩니다.

리눅스 재단 CKAD 시험은 쿠버네티스 기술을 보여줄 수 있는 개발자들에게 훌륭한 기회입니다. 경험이 풍부한 쿠버네티스 전문가이든 초보자이든, 이 자격증은 여러분의 경력을 한 단계 끌어올리고 클라우드 네이티브 개발의 빠르게 변화하는 세계에서 새로운 기회를 열어줄 수 있습니다.

최신 Kubernetes Application Developer CKAD 무료샘플문제 (Q195-Q200):

질문 # 195
You are managing a Kubernetes cluster that runs several microservices. One of these services, called "web-app", needs to have its pods scaled based on the current load. You have created a custom resource called "autoscaling.k8s.i0/v1 alphal" which represents the desired number of replicas for the "web-app" service- Implement a Kubernetes Controller that watches for changes in this custom resource and automatically scales the "web-app" Deployment accordingly.

정답：

설명：

See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create the Custom Resource Definition (CRD):
- Define the custom resource "autoscaling.k8s.io/v1alpha1" using the following YAML file:
- Apply the CRD using 'kubectl apply -f crd.yamr 2. Create the Custom Resource: - Create an instance of the custom resource 'Autoscaling" with the desired number of replicas for the "web-app" Deployment:
- Apply the custom resource using 'kubectl apply -t autoscaling-yamr 3. Create the Kubernetes Controller: - Create a Kubernetes Controller that watches for changes in the "Autoscaling" custom resource and updates the "web-app" Deployment.
4. Deploy the Controller: - Build and deploy tne controller to your Kubernetes cluster. 5. Verify the Controllers Functionality: - Make a change to the "Autoscaling" custom resource (e.g., increase the desired replicas). - Observe that the "web-app" Deployment is automatically scaled based on the new desired replica count. This code implements a basic Kubernetes Controller that monitors the "Autoscaling" custom resource. When the desired replicas are changed, the controller updates the "web-appt' Deployment, ensuring the desired number of replicas is maintained. Note: This example assumes that the "web-app" Deployment exists in the same namespace as the "Autoscaling" custom resource. You might need to adapt the code for different deployments and namespaces.,

질문 # 196
You have a Deployment named 'web-apps that runs 3 replicas of a web application container. This application relies on a database service, also deployed as a Deployment named 'db-service' You need to implement a sidecar pattern using the 'initContainer' feature to ensure that the database service is up and running before the web application container starts. The web application container should only start once the database is reachable.

정답：

설명：
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create the 'initcontainer':
- Add a new container definition within the 'spec.template.spec.initContainerS section of the 'web-app' Deployment.
- Name the container as 'db-checker'
- Specify an image that will be used to perform the database health check. You can use a simple image like 'busybox' and install 'curl' in the 'initContainer' to perform the health check.
- The 'command' for the 'initcontainer' should run a 'while' loop that keeps checking the database service's health endpoint until it returns a successful status code.
- You can define the health endpoint in the 'db-service' Deployment's service definition.
2. Update the 'db-service' Deployment - Ensure that the 'db-servjce' Deployment includes a health cneck endpoint in its service definition. - For example, you can expose a health endpoint at 'http://db-service:5432/health' in your database service's configuration. - The health endpoint should return a successful status code (e.g., 200) if the database is running and ready. - If the database is not reachable, the endpoint should return an error code.
3. Apply the changes: - Apply the updated 'web-app' and 'db-service' Deployment YAML files to your Kubernetes cluster using 'kubectl apply' 4. Verify the implementation: - Once the Deployment iS updated, verity that the 'web-app' pods only start atter the 'db-service' pods are ready. - You can observe the 'initcontainer' logs tor the 'db-checker' container to confirm that the health checks are being performed. - You can also monitor the 'db-service' pods to ensure they are in a 'Running' state. This setup will ensure that the 'web-app' containers will not start until the 'db-service' is reachable and healthy. This ensures that the web application has access to the database and can function properly.,

질문 # 197
You have a Deployment that runs a web application. The application requires a specific version ot a library that is not available in the default container image. How would you use an Init Container to install this library before starting the main application container?

정답：

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

Solution (Step by Step) :
1. Create an Init Container:
- Add an 'initContainerS section to the Deployment's 'spec-template-spec' configuration.
- Define an Init Container with a suitable name (e.g., 'library-installer').
- Specify the image for the Init Container This image should contain the necessary tools and commands to install the required library.
- Replace 'your-library-installer-image:latest with the actual image you want to use.

2. Configure the Main Container: - In the main application container, ensure that the environment variable 'PATH' includes the installation directory of the library installed by the Init Container. - This allows the application to find and use the newly installed library. 3. Apply the Changes: - Apply the updated Deployment configuration using 'kubectl apply -t my-web-app-deployment.yamr. 4. Verify the Installation: - Once the Pods are deployed, you can check the logs of the main application container to confirm that the library is installed and available for use.

질문 # 198
You have a web application that uses two different services: 'frontend' and 'backend'. You want to restrict access to the 'backend' service from all pods except those with the label 'app: frontend'. How would you configure NetworkPolicy to achieve this?

정답：

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

- Replace with your actual namespace. 2. Apply the NetworkPolicy: - Run the following command to apply the NetworkPolicy: bash kubectl apply -f backend-networkpolicy.yaml - This NetworkPolicy defines a policy for pods with the label 'app: backend'. - The 'ingress' rule allows traffic only from pods with the label 'app: frontend'. - All other pods will be blocked from accessing the 'backend' service. This ensures that only the frontend' service can communicate with the 'backend' service. ,

질문 # 199

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.

정답：

설명：
See the solution below.
Explanation
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

질문 # 200
.....

- 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, 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, bbs.t-firefly.com, 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, 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, Disposable vapes