Linux Foundation CKAD Mock Exams | Free Download Test CKAD Centres: Linux Foundation Certified Kubernetes Application Developer Exam

DOWNLOAD the newest ExamsReviews CKAD PDF dumps from Cloud Storage for free: https://drive.google.com/open?id=1IY17ZB07DjHL6owOWLaDGLZvZfoU8fUj

We have confidence and ability to make you get large returns but just need input small investment. our CKAD study materials provide a platform which help you gain knowledge in order to let you outstanding in the labor market and get satisfying job that you like. The content of our CKAD question torrent is easy to master and simplify the important information. It conveys more important information with less answers and questions, thus the learning is easy and efficient. We believe our latest CKAD exam torrent will be the best choice for you.

Our company has always been following the trend of the CKAD certification. Our research and development team not only study what questions will come up in the exam, but also design powerful study tools like CKAD exam simulation software. This Software version of our CKAD learning quesions are famous for its simulating function of the real exam, which can give the candidates a chance to experience the real exam before they really come to it.

>> CKAD Mock Exams <<

Test CKAD Centres - Reliable CKAD Exam Topics

Our CKAD exam braindumps have become a brand that is good enough to stand out in the market. The high quality product like our CKAD study quiz has no need to advertise everywhere, and exerts influential effects which are obvious and everlasting during your preparation. The exam candidates of our CKAD Study Materials are the best living and breathing ads. Just look at the comments on the CKAD training guide, you will know that how popular they are among the candidates.

The CKAD exam is an excellent opportunity for developers who want to demonstrate their expertise in Kubernetes application development. CKAD exam covers a range of topics, including Kubernetes core concepts, pod and service deployment, debugging, troubleshooting, and automation. Candidates who pass the exam will receive a CKAD certification that demonstrates their proficiency in Kubernetes application development. Linux Foundation Certified Kubernetes Application Developer Exam certification is recognized by many organizations in the technology industry and can help developers advance their careers. Additionally, the CKAD certification is a prerequisite for the Certified Kubernetes Administrator (CKA) certification, which is designed for system administrators who manage Kubernetes clusters.

The CKAD Certification Exam is a hands-on, performance-based exam that assesses a candidate's skills in solving real-world problems related to Kubernetes. CKAD exam is conducted on a live Kubernetes cluster, and candidates are required to complete a set of practical tasks within a given time limit. These tasks cover a wide range of topics, including Kubernetes architecture, Kubernetes API objects, pod scheduling, application deployment, and troubleshooting.

Linux Foundation Certified Kubernetes Application Developer Exam Sample Questions (Q114-Q119):

NEW QUESTION # 114
You have a Kubernetes Job that runs a Python script for data processing. The script takes 30 minutes to complete, and you need to ensure that the Job is retried up to 3 times if it fails. Additionally, you want the Job to complete within a maximum of 45 minutes. Create a Job YAML file with appropriate configuration.

Answer:

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create a Job YAML file:
2. Apply the Job YAML file: bash kubectl apply -f data-processing-job.yaml 3. Monitor the Job: bash kubectl get jobs -w This will show the status of the Job, including its completion status and retries, if any. 4. Examine the Job's Pods: bash kubectl get pods -l job-name-data-processing-job You can use the 'kubectl logs command to cneck tne logs of tne POdS created by tne Job to investigate any potential failures. - 'backoffLimit: 3': This specifies that the Job can be retried up to 3 times in case of failures. - 'activeDeadlineSeconds: 2700': This sets the maximum duration for the Job to run (2700 seconds, which is equal to 45 minutes). If the Job exceeds this time limit, it will be automatically terminated. - 'restartPolicy: Never: This ensures that Pods created by the Job will not be restarted automatically. - 'command: ["python", "data_processing_script.py'T: This defines the command to execute inside the container. - 'resources-requests': This defines the minimum resource requirements for the container, including CPU and memory. - 'resources-limits: This can be used to define maximum resource limits for the container. This setup will attempt to run the data processing script If it fails, it will be retried up to 3 times, with an increasing delay between each retry. The Job will be terminated after 45 minutes if it does not complete successfully.,

NEW QUESTION # 115
You have a Deployment named 'database-deployment' that runs a PostgreSQL database container. You want to enforce the following security restrictions:
- The container should only be allowed to run with the I-IID 1000.
- The container should be able to access a specific hostPath volume mounted at '/db-data' for storing database data.
- The container should not be allowed to escalate privileges.
- The container should only have the 'NET BIND SERVICE capability, allowing it to listen on specific ports.
You need to define a SecurityContext in the Deployment configuration to enforce these restrictions.

Answer:

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Define the SecurityContext
- Create a 'securitycontext' section within the 'spec-template-spec-containers' block for your 'database-deployment container-
- Set 'runAsIJsers to '1000' to enforce running as UID 1000.
- Set 'allowPrivilegeEscalation' to 'false' to disable privilege escalation-
- In the 'capabilities' sectiom
- Set 'drop' to an array containing all capabilities except 'NET BIND_SERVICE'
- Set 'add' to an array containing 'NET BIND SERVICE
- Define a 'volumeMount' to mount the '/db-data' hostPath volume.
Solution (Step by Step) :
1. Define the SecurityContext:
- Create a 'securityContext' section within the block for your 'database-deployment container.
- Set 'runAslJser' to "1000' to enforce running as UID 1000.
- Set 'allowPriviIegeEscaIation' to 'false' to disable privilege escalation.
- In the 'capabilities' section:
- Set 'drop' to an array containing all capabilities except 'NET BIND SERVICE
- Set 'add' to an array containing
- Define a 'volumeMount' to mount the '/db-data' hostPath volume.
2. Create the Deployment: - Apply the Deployment YAML file using 'kubectl apply -f database-deployment.yaml. - The 'securityContext' restricts the container's benavior and capabilities. - Setting 'runAslJser' to '1000' forces the container to run as the specified UID. - 'allowPrivilegeEscalation' set to 'false' prevents tne container from gaining higner privileges. - The 'capabilities' section controls specific capabilities. 'drop' removes unwanted capabilities, while 'add' grants specific capabilities. In this case, the

container is allowed to use the capability, enabling it to bind to specific ports. - The 'volumeMount defines the mount point for the hostPath volume, providing access to the specified directory tor database data. This configuration ensures that the 'database-deployment container runs with the specific IJID, cannot escalate privileges, and only has the 'NET BIND SERVICE' capability, while accessing the hostPath volume for database data. This provides a secure environment for your database container.,

NEW QUESTION # 116
You are deploying a microservice application consisting of three components: 'frontend' , 'backend' , and 'database'. You want to ensure that the 'backend' service is deployed only after the 'frontend' service has successfully started and is healthy. Additionally, the 'database' service should be deployed only after the 'backend' service is ready. How would you implement this deployment strategy using Kubernetes deployments?

Answer:

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Define Pre-requisites for Services:
- Create a 'Deployment for each service (frontend', 'backend', and 'database').
- For the 'backend' service, define a 'pre-requisite' in the 'dependencies' section of the 'Deployment' object, specifying that the ' frontend' service needs to be healthy and running. This can be achieved using the 'dependson' field in the 'spec.template.spec_containers' section of the Deployment.
- Similarly, for the 'database service, define a Tre-requisite' specifying that the 'backend' service needs to be healthy and running.
- Example 'frontend' Deployment:
- Example 'backend' Deployment:
- Example 'database' Deployment:
2. Create the Deployments: - Apply the YAML files using 'kubectl apply -f frontend-deployment.yamr , 'kubectl apply -f backend-deployment.yamr , and 'kubectl apply -f database- deployment-yaml. 3. Monitor the Deployment Process: - use 'kubectl get pods -l app=frontend' , 'kubectl get pods -l app=backend' , and 'kubectl get pods -l app=database' to monitor the deployment of the pods. - You will observe that the 'frontend' pods will start first, followed by the 'backend' pods after the 'frontend' pods are healthy. Finally, the 'database' pods will start after the 'backend' pods are healthy. 4. Verify the Deployment Success: - Use 'kubectl describe deployments frontend-deployment', 'kubectl describe deployments backend-deployment , and 'kubectl describe deployments database-deployment' to verify the successful deployment of each service. - Confirm that the 'Ready' status of each pod is true. This strategy ensures that the services are deployed in a predictable and reliable order, ensuring the application's integrity and functionality.,

NEW QUESTION # 117
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 See the solution below.

Answer:

Explanation:
Explanation
Solution:

NEW QUESTION # 118
You have a Deployment for a stateless application that involves several containers. You want to expose this application as a single service using a Service resource- You also want to configure the Service to use a specific label selector to target the correct pods and to ensure that the Service uses a round-robin load balancing strategy for distributing traffic across the pods. Explain how you can create the Service resource and configure it to acnjeve this.

Answer:

Explanation:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create a Service:
- Create a Service resource that defines how the pods of your application will be accessed externallv
- Specify a unique name for the Service and define the port mapping-
- Example:
2. Set Selector: - Use the 'selector field to specify the label that the Service should use to identify the pods it should target - Ensure that the pods of your Deployment are labeled with the specified 'app: myapp' label. 3. Configure Load Balancing: - The default load balancing strategy for Kubernetes Services is round-robin, which distributes traffic evenly across the available pods. - No additional configuration is needed for this strategyc 4. Deploy and Test: - Apply the Service YAML file. - Test the Service by accessing it from outside the cluster. - Ensure that the traffic is distributed evenly across the pods using the round-robin strategy. 5. Optional: External Access: - If you want to expose the Service to the internet, you can set the Service 'type' to 'Load8alancer'. This will create a LoadBalancer resource (often an external IP address) that routes traffic to the Service.

NEW QUESTION # 119
.....

id=1IY17ZB07DjHL6owOWLaDGLZvZfoU8fUj