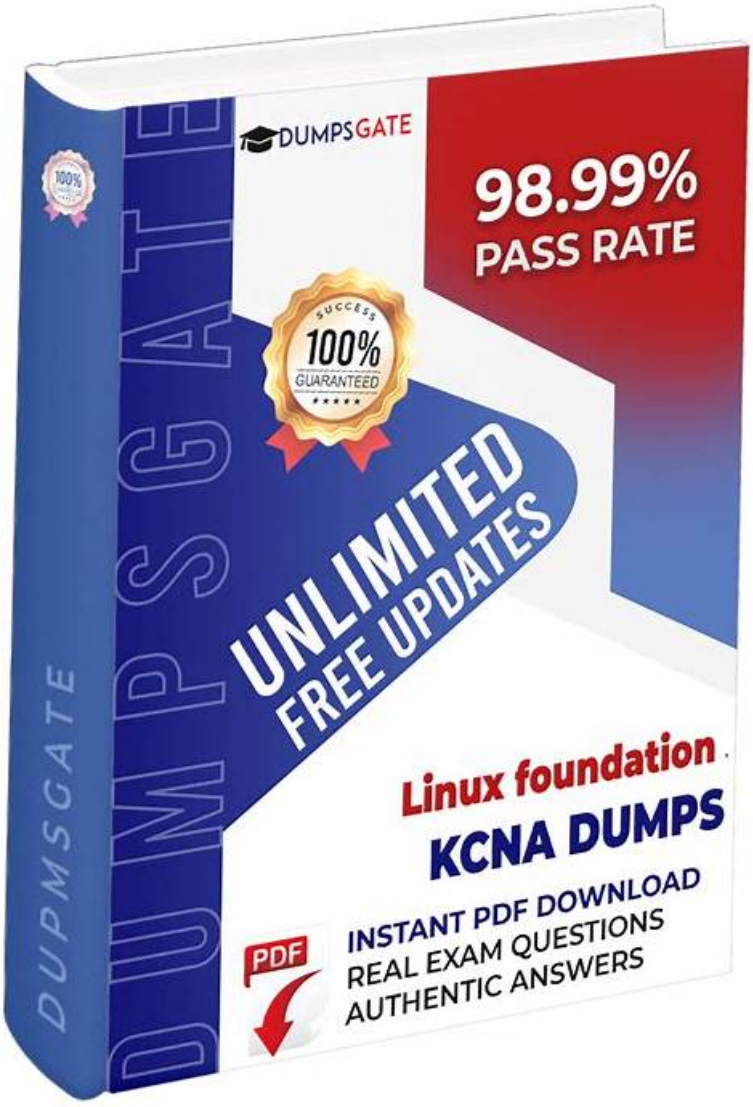
Pass Guaranteed Quiz Linux Foundation - Updated KCNA Complete Exam Dumps



What's more, part of that Actual4test KCNA dumps now are free: https://drive.google.com/open?id=1r7PBsRpUCmH_uMkPyElhbh4dXpSeoOHR

There are three versions of our KCNA study materials so that you can choose the right version for your exam preparation. The test engine is a way of exam simulation that makes you feels the atmosphere of KCNA Real Exam. It brings great convenience for most IT workers because it allows candidates to practice KCNA exam prep anytime and anywhere as long as you download the KCNA dumps pdf.

To earn the KCNA Certification, candidates must pass a rigorous exam that tests their understanding of Kubernetes architecture, deployment, and maintenance. KCNA exam also covers related technologies such as containerization, microservices, and cloud-native application development. Kubernetes and Cloud Native Associate certification is designed for IT professionals who want to demonstrate their expertise in these areas and advance their careers in cloud computing.

>> KCNA Complete Exam Dumps <<

Linux Foundation KCNA Reliable Test Bootcamp & Real KCNA Exam Dumps

Now are you in preparation for KCNA exam? If so, you must be a man with targets. Our Actual4test are committed to help such a man with targets to achieve the goal. KCNA exam simulation software developed by us are filled with the latest and comprehensive questions. If you buy our product, we will offer one year free update of the questions for you. With our software, passing KCNA Exam will no longer be the problem.

The KCNA Certification Exam is an excellent opportunity for IT professionals who want to advance their careers in cloud native computing. Kubernetes and Cloud Native Associate certification is ideal for those who are just starting out in their career, as well as experienced professionals who want to validate their skills in Kubernetes and cloud native computing. Kubernetes and Cloud Native Associate certification is recognized globally and can be a valuable asset to individuals who work in industries that require cloud native computing expertise.

The KCNA exam is a useful credential for individuals who want to work in the field of cloud computing and containerization. It is a vendor-neutral certification that demonstrates a solid understanding of cloud native technologies and their practical applications. Kubernetes and Cloud Native Associate certification exam is open to anyone who has a basic understanding of Linux and cloud computing. It is an online exam that can be taken from anywhere in the world, making it accessible to a global audience. The KCNA Certification is a valuable asset for professionals who want to advance their careers in the cloud computing industry.

Linux Foundation Kubernetes and Cloud Native Associate Sample Questions (Q12-Q17):

NEW QUESTION # 12
How does dynamic storage provisioning work?

- A. An administrator creates a PersistentVolume and includes the name of the PersistentVolume in their Pod YAML definition file.
- B. A user requests dynamically provisioned storage by including an existing StorageClass in their PersistentVolumeClaim.
- C. An administrator creates a StorageClass and includes it in their Pod YAML definition file without creating a PersistentVolumeClaim.
- D. A Pod requests dynamically provisioned storage by including a StorageClass and the Pod name in their PersistentVolumeClaim.

Answer: B

Explanation:
Dynamic provisioning is the Kubernetes mechanism where storage is created on-demand when a user creates a PersistentVolumeClaim (PVC) that references a StorageClass, so A is correct. In this model, the user does not need to pre-create a PersistentVolume (PV). Instead, the StorageClass points to a provisioner (typically a CSI driver) that knows how to create a volume in the underlying storage system (cloud disk, SAN, NAS, etc.). When the PVC is created with storageClassName: <class>, Kubernetes triggers the provisioner to create a new volume and then binds the resulting PV to that PVC.
This is why option B is incorrect: you do not put a StorageClass "in the Pod YAML" to request provisioning. Pods reference PVCs, not StorageClasses directly. Option C is incorrect because the PVC does not need the Pod name; binding is done via the PVC itself. Option D describes static provisioning: an admin pre-creates PVs and users claim them by creating PVCs that match the PV (capacity, access modes, selectors). Static provisioning can work, but it is not dynamic provisioning.
Under the hood, the StorageClass can define parameters like volume type, replication, encryption, and binding behavior (e.g., volumeBindingMode: WaitForFirstConsumer to delay provisioning until the Pod is scheduled, ensuring the volume is created in the correct zone). Reclaim policies (Delete/Retain) define what happens to the underlying volume after the PVC is deleted.
In cloud-native operations, dynamic provisioning is preferred because it improves developer self-service, reduces manual admin work, and makes scaling stateful workloads easier and faster. The essence is: PVC + StorageClass → automatic PV creation and binding.

NEW QUESTION # 13
Your application requires a specific storage class for its persistent dat
a. How do you configure this storage class within your deployment YAML?

- A. Specify the storage class name directly within the 'spec.template.spec.containers[0].volumeMounts[0].name' field of the deployment.
- B. Specify the storage class name within the 'spec.template.spec.containers[01.volumeMounts[0].storageClassName' field of the deployment.
- C. None of the above
- D. Specify the storage class name within the field of the deployment.

- E. Create a separate PersistentVolumeClaim (PVC) with the desired storage class and reference the PVC in the deployment's name' field.

Answer: E

Explanation:
The correct approach is to create a separate PersistentVolumeClaim (PVC) that specifies the desired storage class and reference the PVC in the deployments 'spec_template.spec_containers[0]_volumeMounts[0]_name' field. This ensures the PVC is automatically bound to a PV with the correct storage class. Specifying the storage class name directly within the deployment or the volumeMounts section is not the standard practice for defining storage requirements.

NEW QUESTION # 14
What kind of limitation cgroups allows?

- A. Prioritization
- B. Accounting
- C. Control
- D. Server cpu and memory
- E. None of the options
- F. Resource limiting

Answer: A,B,C,F

NEW QUESTION # 15
You're running a web application on Kubernetes that experiences occasional traffic spikes. Which of the following strategies is most suitable for managing costs during these spikes without compromising performance?

- A. Configure a resource quota for the web application's namespace to limit its resource usage.
- B. Use a horizontal pod autoscaler (HPA) with metrics based on CPU and memory utilization.
- C. Manually scale the deployment up during traffic spikes and down afterward.
- D. Implement a caching layer in front of the web application to reduce the load on the backend servers.
- E. Utilize serverless functions to handle the peak traffic load, automatically scaling as needed.

Answer: B,D

Explanation:
For web applications with unpredictable traffic patterns, dynamic scaling and load optimization are crucial for cost management. Horizontal Pod Autoscaler (HPA) with CPU and memory utilization metrics automatically scales the deployment up and down based on actual load, effectively managing resources during spikes. Implementing a caching layer reduces load on the backend servers, further reducing resource consumption and costs. Manually scaling is inefficient, and resource quotas don't address the dynamic nature of traffic spikes. While serverless functions offer scaling, they might not be suitable for all web application components.

NEW QUESTION # 16
Which kubectl command is useful for collecting information about any type of resource that is active in a Kubernetes cluster?

- A. describe
- B. explain
- C. list
- D. expose

Answer: A

Explanation:
The correct answer is A (describe), used as kubectl describe <resource> <name>. kubectl describe is a troubleshooting-focused command that provides a rich, human-readable view of a specific live object in the cluster, including key fields, status, and-crucially-Events related to that object. This makes it extremely useful for "collecting information" about almost any active resource: Pods, Deployments, Nodes, Services, PersistentVolumeClaims, and more.
kubectl get (not listed) is typically used for listing objects and their summary fields, but kubectl describe goes deeper: for a Pod it will

show container images, resource requests/limits, probes, mounted volumes, node assignment, IPs, conditions, and recent scheduling/pulling/starting events. For a Node it shows capacity/allocatable resources, labels/taints, conditions, and node events. Those event details often explain why something is Pending, failing to pull images, failing readiness checks, or being evicted.
Option B ("list") is not a standard kubectl subcommand for retrieving resource information (you would use get for listing). Option C (expose) is for creating a Service to expose a resource (like a Deployment). Option D (explain) is for viewing API schema/field documentation (e.g., kubectl explain deployment.spec.replicas) and does not report what is currently happening in the cluster.
So, for gathering detailed live diagnostics about a resource in the cluster, the best kubectl command is kubectl describe, which corresponds to option A.

NEW QUESTION # 17
.....