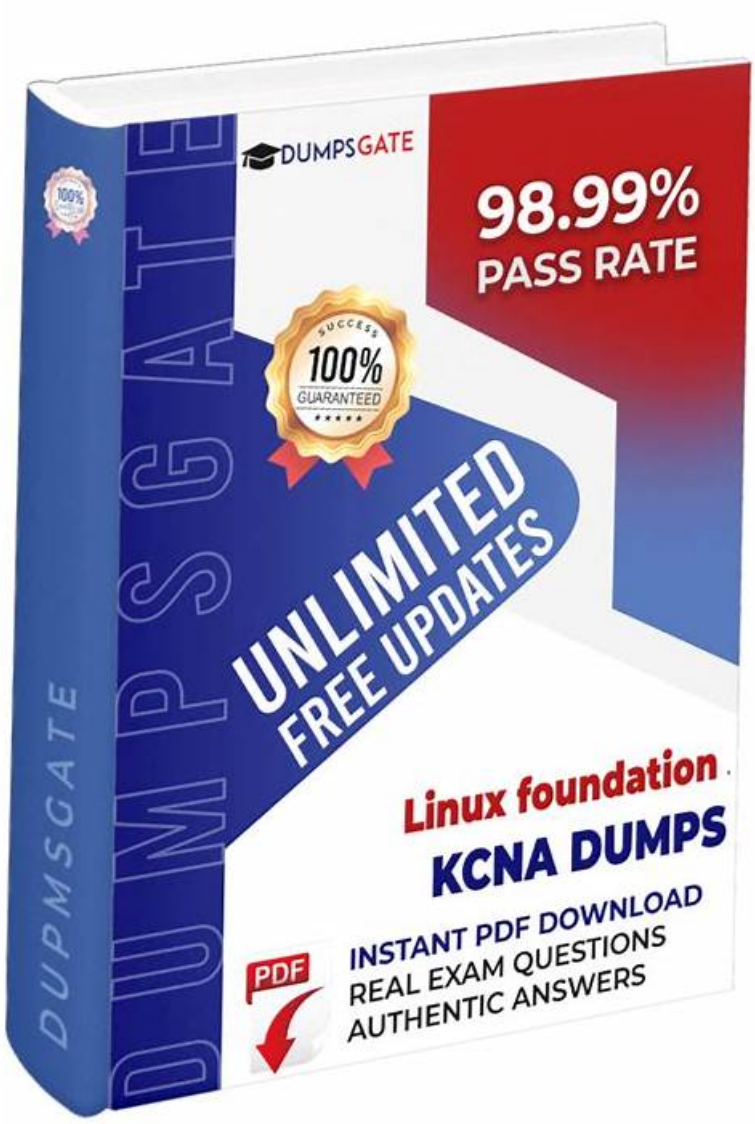
New KCNA Dumps Files, Free KCNA Brain Dumps



2026 Latest DumpsActual KCNA PDF Dumps and KCNA Exam Engine Free Share: https://drive.google.com/open?id=10kIWQ97DbBDnR5n4MB-_3keFvJ5sEXK0

By adding all important points into practice materials with attached services supporting your access of the newest and trendiest knowledge, our KCNA preparation materials are quite suitable for you right now as long as you want to pass the KCNA exam as soon as possible and with a 100% pass guarantee. Our KCNA study questions are so popular that everyday there are numerous of our loyal customers wrote to inform and thank us that they passed their exams for our exam braindumps.

Linux Foundation KCNA (Kubernetes and Cloud Native Associate) Exam is a certification program designed for IT professionals who want to demonstrate their expertise in cloud-native technologies and Kubernetes. KCNA exam covers a broad range of topics, including containerization, Kubernetes architecture, and deployment, as well as cloud-native application development and management. Kubernetes and Cloud Native Associate certification is recognized globally and is highly valued by employers who are looking for skilled professionals in this rapidly growing field.

>> New KCNA Dumps Files <<

Free Updates for 365 Days on Linux Foundation KCNA Exam Questions

Our KCNA training materials are compiled by professional experts. All the necessary points have been mentioned in our KCNA

practice engine particularly. About some tough questions or important points, they left notes under them. Besides, our experts will concern about changes happened in KCNA study prep all the time. Provided you have a strong determination, as well as the help of our KCNA learning guide, you can have success absolutely.

The Kubernetes and Cloud Native Associate (KCNA) certification exam is offered by the Linux Foundation, a non-profit organization that aims to promote and support open source technology. Kubernetes and Cloud Native Associate certification exam is designed to test the knowledge and skills of individuals who are interested in working with Kubernetes and cloud native technologies. KCNA Exam covers a range of topics, including containerization, Kubernetes architecture, deployment, and management.

Linux Foundation Kubernetes and Cloud Native Associate Sample Questions (Q88-Q93):

NEW QUESTION # 88
Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called:

- A. Containers
- B. Namespaces
- C. Hypervisors
- D. cgroups

Answer: B

Explanation:
Kubernetes provides "virtual clusters" within a single physical cluster primarily through Namespaces, so A is correct. Namespaces are a logical partitioning mechanism that scopes many Kubernetes resources (Pods, Services, Deployments, ConfigMaps, Secrets, etc.) into separate environments. This enables multiple teams, applications, or environments (dev/test/prod) to share a cluster while keeping their resource names and access controls separated.
Namespaces are often described as "soft multi-tenancy." They don't provide full isolation like separate clusters, but they do allow administrators to apply controls per namespace:
RBAC rules can grant different permissions per namespace (who can read Secrets, who can deploy workloads, etc.).
ResourceQuotas and LimitRanges can enforce fair usage and prevent one namespace from consuming all cluster resources.
NetworkPolicies can isolate traffic between namespaces (depending on the CNI).
Containers are runtime units inside Pods and are not "virtual clusters." Hypervisors are virtualization components for VMs, not Kubernetes partitioning constructs. cgroups are Linux kernel primitives for resource control, not Kubernetes virtual cluster constructs.
While there are other "virtual cluster" approaches (like vcluster projects) that create stronger virtualized control planes, the built-in Kubernetes mechanism referenced by this question is namespaces. Therefore, the correct answer is A: Namespaces.

NEW QUESTION # 89
What is the main purpose of the Open Container Initiative (OCI)?

- A. Creating open industry standards around container formats and runtimes.
- B. Improving the security of standards around container formats and runtimes.
- C. Creating industry standards around container formats and runtimes for private purposes.
- D. Accelerating the adoption of containers and Kubernetes in the industry.

Answer: A

Explanation:
B is correct: the OCI's main purpose is to create open, vendor-neutral industry standards for container image formats and container runtimes. Standardization is critical in container orchestration because portability is a core promise: you should be able to build an image once and run it across different environments and runtimes without rewriting packaging or execution logic.
OCI defines (at a high level) two foundational specs:
Image specification: how container images are packaged (layers, metadata, manifests).
Runtime specification: how to run a container (filesystem setup, namespaces/cgroups behavior, lifecycle).
These standards enable interoperability across tooling. For example, higher-level runtimes (like containerd or CRI-O) rely on OCI-compliant components (often runc or equivalents) to execute containers consistently.
Why the other options are not the best answer:
A (accelerating adoption) might be an indirect outcome, but it's not the OCI's core charter.

C is contradictory ("industry standards" but "for private purposes")-OCI is explicitly about open standards.
D (improving security) can be helped by standardization and best practices, but OCI is not primarily a security standards body; its central function is format and runtime interoperability.
In Kubernetes specifically, OCI is part of the "plumbing" that makes runtimes replaceable. Kubernetes talks to runtimes via CRI; runtimes execute containers via OCI. This layering helps Kubernetes remain runtime-agnostic while still benefiting from consistent container behavior everywhere.
Therefore, the correct choice is B: OCI creates open standards around container formats and runtimes.

NEW QUESTION # 90
What is CloudEvents?

- A. It is a specification for describing event data in common formats in all cloud providers including major cloud providers.
- B. It is a Kubernetes specification for describing events data in common formats for iCloud services, iOS platforms and iMac.
- C. It is a specification for describing event data in common formats for Kubernetes network traffic management and cloud providers.
- D. It is a specification for describing event data in common formats to provide interoperability across services, platforms and systems.

Answer: D

Explanation:
CloudEvents is an open specification for describing event data in a common way to enable interoperability across services, platforms, and systems, so C is correct. In cloud-native architectures, many components communicate asynchronously via events (message brokers, event buses, webhooks). Without a standard envelope, each producer and consumer invents its own event structure, making integration brittle. CloudEvents addresses this by standardizing core metadata fields-like event id, source, type, specversion, and time-and defining how event payloads are carried.
This helps systems interoperate regardless of transport. CloudEvents can be serialized as JSON or other encodings and carried over HTTP, messaging systems, or other protocols. By using a shared spec, you can route, filter, validate, and transform events more consistently.
Option A is too narrow and incorrectly ties CloudEvents to Kubernetes traffic management; CloudEvents is broader than Kubernetes. Option B is closer but still framed incorrectly-CloudEvents is not merely "for all cloud providers," it is an interoperability spec across services and platforms, including but not limited to cloud provider event systems. Option D is clearly incorrect.
In Kubernetes ecosystems, CloudEvents is relevant to event-driven systems and serverless platforms (e.g., Knative Eventing and other eventing frameworks) because it provides a consistent event contract across producers and consumers. That consistency reduces coupling, supports better tooling (schema validation, tracing correlation), and makes event-driven architectures easier to operate at scale.
So, the correct definition is C: a specification for common event formats to enable interoperability across systems.

NEW QUESTION # 91
You have a Kubernetes cluster running on AWS. You want to configure a persistent volume claim (PVC) that uses an AWS EBS volume for storage. Which annotation can be used to specify the EBS volume type?

- A. volume.beta.kubernetes.io/aws-ebs-volume-size
- B. volume.beta.kubernetes.io/aws-ebs-volume-encrypted
- C. volume.beta.kubernetes.io/aws-ebs-volume-type
- D. volume.beta.kubernetes.io/storage-class
- E. volume.beta.kubernetes.io/storage-provisioner

Answer: C

Explanation:
The annotation •volume.beta.kubernetes.io/aws-ebs-volume-type• is used to specify the EBS volume type (e.g., 'gp2 , 701', 'standard') when using an AWS EBS volume for persistent storage. Option 'A' is used to specify the storage class for the PVC. Option 'B' specifies the storage provisioner, which is responsible for creating the volume. Option 'D' is used to specify the size of the EBS volume. Option 'E' is for specifying whether the EBS volume should be encrypted.

NEW QUESTION # 92

What is the main difference between Argo vs. Flux CD?

- A. Argo is push-based, and Flux is pull-based
- B. No difference; both are push-based
- C. Argo is pull-based, and Flux is push-based
- D. No difference; both are pull-based

Answer: A

Explanation:
AgroCD: https://argo-cd.readthedocs.io/en/stable/developer-guide/ci/#can-i-retrigger-the-checks-without-pushing-a-new-commit
FluxCD: https://fluxcd.io/

NEW QUESTION # 93
.....