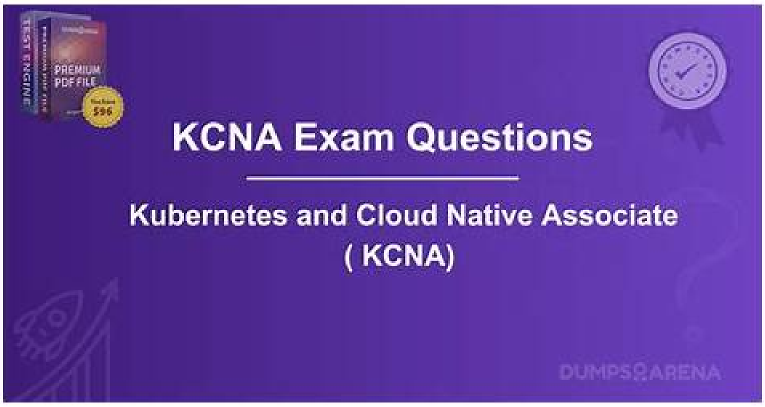
KCNA PDF Question & KCNA Pdf Torrent



BTW, DOWNLOAD part of TestkingPDF KCNA dumps from Cloud Storage: https://drive.google.com/open?id=1IxUTd2lX1gZJGUk0V1Ym9vN9Tz3ZvrUI

Full refund is available if you fail to pass the exam in your first attempt after buying KCNA exam bootcamp from us, and we will refund your money, In addition, KCNA exam dumps contain both questions and answers, and it's convenient for you to check the answers after practicing. KCNA exam botcamp cover most of the knowledge points of the exam, and you can master the major knowledge points as well as improve your professional ability in the process of training. We have online and offline chat service for KCNA Exam Dumps, and if you have any questions, you can consult us.

To prepare for the KCNA certification exam, candidates can take advantage of a variety of resources provided by the Linux Foundation. These include online training courses, study guides, and practice exams. Additionally, candidates can participate in community forums and discussion groups to gain insights and advice from other professionals in the field.

The Linux Foundation KCNA exam is designed to test the candidate's knowledge and understanding of Kubernetes and cloud-native technologies. It covers a range of topics such as deploying, managing, and scaling applications in Kubernetes, understanding Kubernetes architecture and components, and configuring Kubernetes networking and security. KCNA Exam also tests the candidate's knowledge of containerization technologies such as Docker and container orchestration tools like Helm.

>> KCNA PDF Question <<

Pass Guaranteed 2026 High-quality KCNA: Kubernetes and Cloud Native Associate PDF Question

With the rapid development of the economy, the demands of society on us are getting higher and higher. If you can have KCNA certification, then you will be more competitive in society. Our study materials will help you get the according certification you want to have. Believe me, after using our study materials, you will improve your work efficiency. You will get more opportunities than others, and your dreams may really come true in the near future. KCNA Test Guide will make you more prominent in the labor market than others, and more opportunities will take the initiative to find you.

Linux Foundation KCNA Exam is a certification program that validates an individual's knowledge of Kubernetes and cloud-native technologies. KCNA exam is designed to be challenging and is suitable for IT professionals who are interested in working with these technologies. Kubernetes and Cloud Native Associate certification is recognized by leading companies in the tech industry and can help individuals advance their career and increase their earning potential. Candidates are encouraged to prepare thoroughly before attempting the exam to ensure success.

Linux Foundation Kubernetes and Cloud Native Associate Sample Questions (Q100-Q105):

NEW QUESTION # 100
Which of the following is a lightweight tool that manages traffic flows between services, enforces access policies, and aggregates telemetry data, all without requiring changes to application code?

- A. Linkerd

- B. NetworkPolicy
- C. kube-proxy
- D. Nginx

Answer: A

Explanation:
Linkerd is a lightweight service mesh that manages service-to-service traffic, security policies, and telemetry without requiring application code changes-so B is correct. A service mesh introduces a dedicated layer for east-west traffic (internal service calls) and typically provides features like mutual TLS (mTLS), retries /timeouts, traffic shaping, and consistent metrics/tracing signals. Linkerd is known for being simpler and resource-efficient relative to some alternatives, which aligns with the "lightweight tool" phrasing.
Why this matches the description: In a service mesh, workload traffic is intercepted by a proxy layer (often as a sidecar or node-level/ambient proxy) and managed centrally by mesh control components. This allows security and traffic policy to be applied uniformly without modifying each microservice. Telemetry is also generated consistently because the proxies observe traffic directly and emit metrics and traces about request rates, latency, and errors.
The other choices don't fit. NetworkPolicy is a Kubernetes resource that controls allowed network flows (L3 /L4) but does not provide L7 traffic management, retries, identity-based mTLS, or automatic telemetry aggregation. kube-proxy implements Service networking rules (ClusterIP/NodePort forwarding) but does not enforce access policies at the service identity level and is not a telemetry system. Nginx can be used as an ingress controller or reverse proxy, but it is not inherently a full service mesh spanning all service-to-service communication and policy/telemetry across the mesh by default.
In cloud native architecture, service meshes help address cross-cutting concerns-security, observability, and traffic management-without embedding that logic into every application. The question's combination of "traffic flows," "access policies," and "aggregates telemetry" maps directly to a mesh, and the lightweight mesh option provided is Linkerd.

NEW QUESTION # 101
Which of the following systems is NOT compatible with the CRI runtime interface standard?
(Typo corrected: "CRI-0" → "CRI-O")

- A. systemd
- B. CRI-O
- C. containerd
- D. dockershim

Answer: A

Explanation:
Kubernetes uses the Container Runtime Interface (CRI) to support pluggable container runtimes. The kubelet talks to a CRI-compatible runtime via gRPC, and that runtime is responsible for pulling images and running containers. In this context, containerd and CRI-O are CRI-compatible container runtimes (or runtime stacks) used widely with Kubernetes, and dockershim historically served as a compatibility layer that allowed kubelet to talk to Docker Engine as if it were CRI (before dockershim was removed from kubelet in newer Kubernetes versions). That leaves systemd as the correct "NOT compatible with CRI" answer, so C is correct.
systemd is an init system and service manager for Linux. While it can be involved in how services (like kubelet) are started and managed on the host, it is not a container runtime implementing CRI. It does not provide CRI gRPC endpoints for kubelet, nor does it manage containers in the CRI sense.
The deeper Kubernetes concept here is separation of responsibilities: kubelet is responsible for Pod lifecycle at the node level, but it delegates "run containers" to a runtime via CRI. Runtimes like containerd and CRI-O implement that contract; Kubernetes can swap them without changing kubelet logic. Historically, dockershim translated kubelet's CRI calls into Docker Engine calls. Even though dockershim is no longer part of kubelet, it was still "CRI-adjacent" in purpose and often treated as compatible in older curricula. Therefore, among the provided options, systemd is the only one that is clearly not a CRI-compatible runtime system, making C correct.

NEW QUESTION # 102
In Kubernetes, what is the primary purpose of creating a Service resource for a Deployment?

- A. To automatically adjust the number of Pods based on CPU or memory utilization metrics.

- B. To centrally manage and apply runtime configuration values for application components.
- C. To provide a stable endpoint for accessing Pods even when their IP addresses change.
- D. To define and attach persistent volumes that store application data across Pod restarts.

Answer: C

Explanation:
In Kubernetes, Pods are inherently ephemeral. They can be created, destroyed, restarted, or rescheduled at any time, and each time this happens, a Pod may receive a new IP address. This dynamic behavior is essential for resilience and scalability, but it also creates a challenge for reliably accessing application workloads. The Service resource addresses this problem by providing a stable network endpoint for a group of Pods, making option B the correct answer.
A Service selects Pods using label selectors-typically the same labels applied by a Deployment-and exposes them through a consistent virtual IP address (ClusterIP) and DNS name. Regardless of how many Pods are running or whether individual Pods are replaced, the Service remains stable and automatically routes traffic to healthy Pods. This abstraction allows clients to communicate with an application without needing to track individual Pod IPs.
Deployments are responsible for managing the lifecycle of Pods, including scaling, rolling updates, and self- healing. However, Deployments do not provide networking or service discovery capabilities. Without a Service, consumers would need to directly reference Pod IPs, which would break as soon as Pods are rescheduled or updated.
Option A is incorrect because centralized configuration management is handled using ConfigMaps and Secrets, not Services. Option C is incorrect because automatic scaling based on CPU or memory is the responsibility of the Horizontal Pod Autoscaler (HPA), not Services. Option D is incorrect because persistent storage is managed using PersistentVolume and PersistentVolumeClaim resources, which are unrelated to Services.
Services can be configured for different access patterns, such as ClusterIP for internal communication, NodePort or LoadBalancer for external access, and headless Services for direct Pod discovery. Despite these variations, their core purpose remains the same: providing a reliable and stable way to access Pods managed by a Deployment.
Therefore, the correct and verified answer is Option B, which aligns with Kubernetes networking fundamentals and official documentation.

NEW QUESTION # 103
Kubernetes ___ protect you against voluntary interruptions (such as deleting Pods, draining nodes) to run applications in a highly available manner.

- A. Pod Topology Spread Constraints
- B. Resource Limits and Requests
- C. Taints and Tolerations
- D. Pod Disruption Budgets

Answer: D

Explanation:
The correct answer is B: Pod Disruption Budgets (PDBs). A PDB is a policy object that limits how many Pods of an application can be voluntarily disrupted at the same time. "Voluntary disruptions" include actions such as draining a node for maintenance (kubectl drain), cluster upgrades, or an administrator deleting Pods.
The core purpose is to preserve availability by ensuring that a minimum number (or percentage) of replicas remain running and ready while those planned disruptions occur.
A PDB is typically defined with either minAvailable (e.g., "at least 3 Pods must remain available") or maxUnavailable (e.g., "no more than 1 Pod can be unavailable"). Kubernetes uses this budget when performing eviction operations. If evicting a Pod would violate the PDB, the eviction is blocked (or delayed), which forces maintenance workflows to proceed more safely-either by draining more slowly, scaling up first, or scheduling maintenance in stages.
Why the other options are not correct: topology spread constraints (A) influence scheduling distribution across failure domains but don't directly protect against voluntary disruptions. Taints and tolerations (C) control where Pods can schedule, not how many can be disrupted. Resource requests/limits (D) control CPU
/memory allocation and do not guard availability during drains or deletions.
PDBs also work best when paired with Deployments/StatefulSets that maintain replicas and with readiness probes that accurately represent whether a Pod can serve traffic. PDBs do not prevent involuntary disruptions (node crashes), but they materially reduce risk during planned operations-exactly what the question is targeting.

NEW QUESTION # 104

Which are the core features provided by a service mesh?

- A. Security vulnerability scanning
- B. Configuration management
- C. Distributing and replicating data
- D. Authentication and authorization

Answer: D

Explanation:
A is the correct answer because a service mesh primarily focuses on securing and managing service-to- service communication, and a core part of that is authentication and authorization. In microservices architectures, internal ("east-west") traffic can become a complex web of calls. A service mesh introduces a dedicated communication layer-commonly implemented with sidecar proxies or node proxies plus a control plane-to apply consistent security and traffic policies across services.
Authentication in a mesh typically means service identity: each workload gets an identity (often via certificates), enabling mutual TLS (mTLS) so services can verify each other and encrypt traffic in transit.
Authorization then builds on identity to enforce "who can talk to whom" via policies (for example: service A can call service B only on certain paths or methods). These capabilities are central because they reduce the need for every development team to implement and maintain custom security libraries correctly.
Why the other answers are incorrect:
* B (data distribution/replication) is a storage/database concern, not a mesh function.
* C (vulnerability scanning) is typically part of CI/CD and supply-chain security tooling, not service-to- service runtime traffic management.
* D (configuration management) is broader (GitOps, IaC, Helm/Kustomize); a mesh does have configuration, but "configuration management" is not the defining core feature tested here.
Service meshes also commonly provide traffic management (timeouts, retries, circuit breaking, canary routing) and telemetry (metrics/traces), but among the listed options, authentication and authorization best matches "core features." It captures the mesh's role in standardizing secure communications in a distributed system.
So, the verified correct answer is A.

NEW QUESTION # 105
......

KCNA Pdf Torrent: https://www.testkingpdf.com/KCNA-testking-pdf-torrent.html