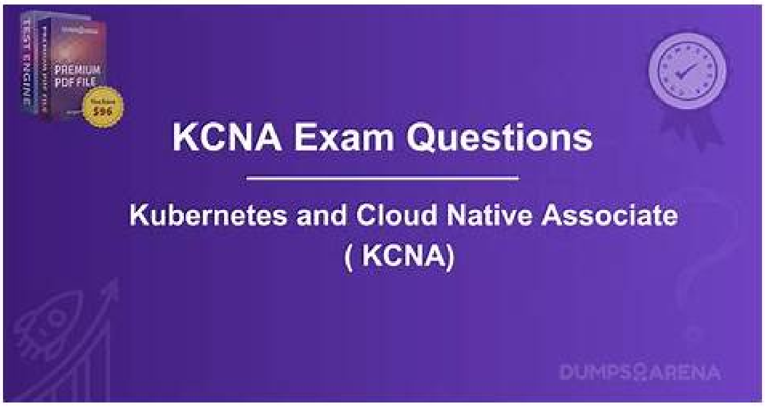
KCNA Valid Mock Exam | Real KCNA Exam Questions



2026 Latest PrepAwayPDF KCNA PDF Dumps and KCNA Exam Engine Free Share: https://drive.google.com/open?id=1yRS6fbZ5IdbiGxwaPwVj1DA8A-uiWD43

Are you still looking for KCNA exam materials? Don't worry about it, because you find us, which means that you've found a shortcut to pass KCNA certification exam. With research and development of IT certification test software for years, our PrepAwayPDF team had a very good reputation in the world. We provide the most comprehensive and effective help to those who are preparing for the important exams such as KCNA Exam.

The KCNA Exam is a vendor-neutral certification, which means that it is not tied to any specific vendor or technology. This makes it an excellent choice for professionals who are looking to broaden their knowledge and skills in cloud-native computing, and who want to demonstrate their expertise to potential employers.

>> KCNA Valid Mock Exam <<

Real KCNA Exam Questions | Exam Discount KCNA Voucher

Let me introduce our KCNA study guide to you in some aspects. First of all, there are three versions of KCNA guide quiz. You can choose the most suitable version based on your own schedule. PC version, PDF version and APP version, these three versions of KCNA Exam Materials have their own characteristics you can definitely find the right one for you. Secondly, you can find that our price of the KCNA learning braindumps is quite favorable. And some times, we will give discounts for them.

Linux Foundation Kubernetes and Cloud Native Associate Sample Questions (Q24-Q29):

NEW QUESTION # 24
How is application data maintained in containers?

- A. Store data into volumes.
- B. Store data into sidecar containers.
- C. Store data into data folders.
- D. Store data in separate folders.

Answer: A

Explanation:
Container filesystems are ephemeral: the writable layer is tied to the container lifecycle and can be lost when containers are recreated. Therefore, maintaining application data correctly means storing it in volumes, making D the correct answer. In Kubernetes, volumes provide durable or shareable storage that is mounted into containers at specific paths. Depending on the volume type, the data can persist across container restarts and even Pod rescheduling.
Kubernetes supports many volume patterns. For transient scratch data you might use emptyDir (ephemeral for the Pod's lifetime). For durable state, you typically use PersistentVolumes consumed by PersistentVolumeClaims (PVCs), backed by storage systems via CSI drivers (cloud disks, SAN/NAS, distributed storage). This decouples the application container image from its state and enables rolling updates, rescheduling, and scaling without losing data.

Options A and B ("folders") are incomplete because folders inside the container filesystem do not guarantee persistence. A folder is only as durable as the underlying storage; without a mounted volume, it lives in the container's writable layer and will disappear when the container is replaced. Option C is incorrect because
"sidecar containers" are not a data durability mechanism; sidecars can help ship logs or sync data, but persistent data should still be stored on volumes (or external services like managed databases).
From an application delivery standpoint, the principle is: containers should be immutable and disposable, and state should be externalized. Volumes (and external managed services) make this possible. In Kubernetes, this is a foundational pattern enabling safe rollouts, self-healing, and portability: the platform can kill and recreate Pods freely because data is maintained independently via volumes.
Therefore, the verified correct choice is D: Store data into volumes.

NEW QUESTION # 25
What is a Service?

- A. An NGINX load balancer that gets deployed for an application.
- B. The network configuration for a group of Pods.
- C. A way to expose an application running on a set of Pods.
- D. A static network mapping from a Pod to a port.

Answer: C

Explanation:
The correct answer is B: a Kubernetes Service is a stable way to expose an application running on a set of Pods. Pods are ephemeral-IPs can change when Pods are recreated, rescheduled, or scaled. A Service provides a consistent network identity (DNS name and usually a ClusterIP virtual IP) and a policy for routing traffic to the current healthy backends.
Typically, a Service uses a label selector to determine which Pods are part of the backend set. Kubernetes then maintains the corresponding endpoint data (Endpoints/EndpointSlice), and the cluster dataplane (kube- proxy or an eBPF-based implementation) forwards traffic from the Service IP/port to one of the Pod IPs. This enables reliable service discovery and load distribution across replicas, especially during rolling updates where Pods are constantly replaced.
Option A is incorrect because Service routing is not a "static mapping from a Pod to a port." It's dynamic and targets a set of Pods. Option C is too vague and misstates the concept; while Services relate to networking, they are not "the network configuration for a group of Pods" (that's closer to NetworkPolicy/CNI configuration). Option D is incorrect because Kubernetes does not automatically deploy an NGINX load balancer when you create a Service. NGINX might be used as an Ingress controller or external load balancer in some setups, but a Service is a Kubernetes API abstraction, not a specific NGINX component.
Services come in several types (ClusterIP, NodePort, LoadBalancer, ExternalName), but the core definition remains the same: stable access to a dynamic set of Pods. This is foundational for microservices and for decoupling clients from the churn of Pod lifecycles.
So, the verified correct definition is B.

NEW QUESTION # 26
What helps an organization to deliver software more securely at a higher velocity?

- A. Kubernetes
- B. Docker Images
- C. CI/CD Pipeline
- D. apt-get

Answer: C

Explanation:
A CI/CD pipeline is a core practice/tooling approach that enables organizations to deliver software faster and more securely, so D is correct. CI (Continuous Integration) automates building and testing code changes frequently, reducing integration risk and catching defects early. CD (Continuous Delivery/Deployment) automates releasing validated builds into environments using consistent, repeatable steps-reducing manual errors and enabling rapid iteration.
Security improves because automation enables standardized checks on every change: static analysis, dependency scanning, container image scanning, policy validation, and signing/verification steps can be integrated into the pipeline. Instead of relying on ad-hoc human processes, security controls become repeatable gates. In Kubernetes environments, pipelines commonly build container

images, run tests, publish artifacts to registries, and then deploy via manifests, Helm, or GitOps controllers-keeping deployments consistent and auditable.
Option A (Kubernetes) is a platform that helps run and manage workloads, but by itself it doesn't guarantee secure high-velocity delivery. It provides primitives (rollouts, declarative config, RBAC), yet the delivery workflow still needs automation. Option B (apt-get) is a package manager for Debian-based systems and is not a delivery pipeline. Option C (Docker Images) are artifacts; they improve portability and repeatability, but they don't provide the end-to-end automation of building, testing, promoting, and deploying across environments.
In cloud-native application delivery, the pipeline is the "engine" that turns code changes into safe production releases. Combined with Kubernetes' declarative deployment model (Deployments, rolling updates, health probes), a CI/CD pipeline supports frequent releases with controlled rollouts, fast rollback, and strong auditability. That is exactly what the question is targeting. Therefore, the verified answer is D.

NEW QUESTION # 27
Which type of Service requires manual creation of Endpoints?

- A. NodePort
- B. Services without selectors
- C. LoadBalancer
- D. ClusterIP with selectors

Answer: B

Explanation:
A Kubernetes Service without selectors requires you to manage its backend endpoints manually, so B is correct. Normally, a Service uses a selector to match a set of Pods (by labels). Kubernetes then automatically maintains the backend list (historically Endpoints, now commonly EndpointSlice) by tracking which Pods match the selector and are Ready. This automation is one of the key reasons Services provide stable connectivity to dynamic Pods.
When you create a Service without a selector, Kubernetes has no way to know which Pods (or external IPs) should receive traffic. In that pattern, you explicitly create an Endpoints object (or EndpointSlices, depending on your approach and controller support) that maps the Service name to one or more IP:port tuples. This is commonly used to represent external services (e.g., a database running outside the cluster) while still providing a stable Kubernetes Service DNS name for in-cluster clients. Another use case is advanced migration scenarios where endpoints are controlled by custom controllers rather than label selection.
Why the other options are wrong: Service types like ClusterIP, NodePort, and LoadBalancer describe how a Service is exposed, but they do not inherently require manual endpoint management. A ClusterIP Service with selectors (D) is the standard case where endpoints are automatically created and updated. NodePort and LoadBalancer Services also typically use selectors and therefore inherit automatic endpoint management; the difference is in how traffic enters the cluster, not how backends are discovered.
Operationally, when using Services without selectors, you must ensure endpoint IPs remain correct, health is accounted for (often via external tooling), and you update endpoints when backends change. The key concept is: no selector # Kubernetes can't auto-populate endpoints # you must provide them.

NEW QUESTION # 28
How many different Kubernetes service types can you define?

- A. 0
- B. 1
- C. 2
- D. 3

Answer: A

Explanation:
Kubernetes defines four primary Service types, which is why C (4) is correct. The commonly recognized Service spec.type values are:
* ClusterIP: The default type. Exposes the Service on an internal virtual IP reachable only within the cluster. This supports typical east-west traffic between workloads.
* NodePort: Exposes the Service on a static port on each node. Traffic to <NodeIP>:<NodePort> is forwarded to the Service endpoints. This is often used for simple external access in environments without load balancers, or as a building block for other

systems.
* LoadBalancer: Integrates with a cloud provider (or load balancer implementation) to provision an external load balancer and route traffic to the Service. This is common in managed Kubernetes.
* ExternalName: Maps the Service name to an external DNS name via a CNAME record, allowing in- cluster clients to use a consistent Service DNS name to reach an external dependency.
Some people also talk about "Headless Services," but headless is not a separate type; it's a behavior achieved by setting clusterIP: None. Headless Services still use the Service API object but change DNS and virtual-IP behavior to return endpoint IPs directly rather than a ClusterIP. That's why the canonical count of "Service types" is four.
This question tests understanding of the Service abstraction: Service type controls how a stable service identity is exposed (internal VIP, node port, external LB, or DNS alias), while selectors/endpoints control where traffic goes (the backend Pods). Different environments will favor different types: ClusterIP for internal microservices, LoadBalancer for external exposure in cloud, NodePort for bare-metal or simple access, ExternalName for bridging to outside services.
Therefore, the verified answer is C (4).

NEW QUESTION # 29
......

Our KCNA study materials will be very useful for all people to improve their learning efficiency. If you do all things with efficient, you will have a promotion easily. If you want to spend less time on preparing for your KCNA exam, if you want to pass your KCNA exam and get the certification in a short time, our KCNA Study Materials will be your best choice to help you achieve your dream. Only studing with our KCNA exam questions for 20 to 30 hours, you will be able to pass the KCNA exam with confidence.