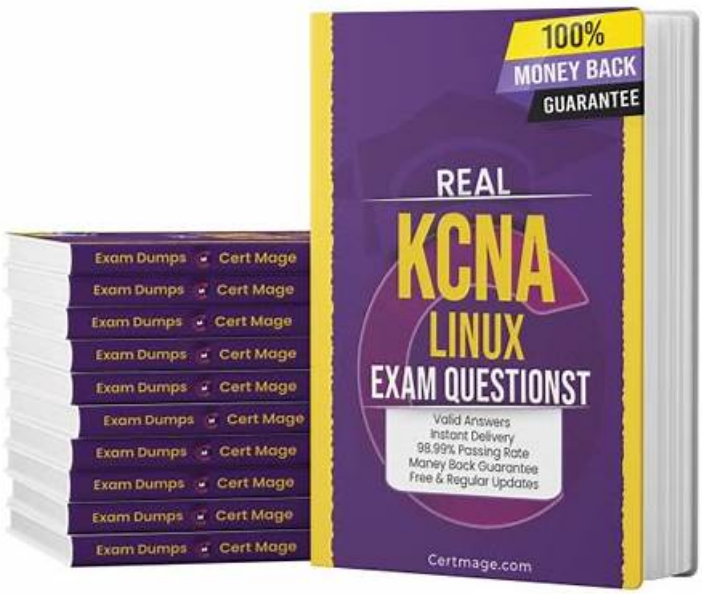
Reliable KCNA Test Bootcamp - Reliable KCNA Test Dumps



BONUS!!! Download part of Prep4SureReview KCNA dumps for free: https://drive.google.com/open?id=1ROyKVbRu-GeP8EZaqeIMG4FMhE4_tumr

Do you want to pass the exam just for one time? If you do want choose our KCNA exam dumps. The pass rate is 98%, and pass guarantee and money back guarantee ig f you fail to pass the exam .Besides we also have the free demo for you to try, before buying, it will help you to have a general idea of the KCNA Exam Dumps. If you have any questions, please contact us directly, we will try our best to help you the problem, so don't hesitate to contact us.

Linux Foundation KCNA certification is a valuable credential for IT professionals who want to demonstrate their expertise in cloud-native technologies. With a wide range of study resources available and a vendor-neutral approach, the exam is an attractive option for professionals who work with different cloud platforms and want to showcase their skills in a way that is recognized across the industry.

To prepare for the KCNA exam, candidates can take advantage of various online courses and resources provided by the Linux Foundation. These resources cover the exam topics in-depth and also provide hands-on experience with tools and technologies used in the industry. Passing the KCNA Exam requires a deep understanding of Kubernetes and Cloud Native technologies and their practical applications. Upon successful completion of the exam, candidates receive a digital certification that can be shared on their professional profiles, proving their expertise in the field of cloud computing.

>> Reliable KCNA Test Bootcamp <<

2026 Newest KCNA: Reliable Kubernetes and Cloud Native Associate Test Bootcamp

We are a team of certified professionals with lots of experience in editing KCNA exam questions. Every candidate should have more than 11 years' education experience in this filed of KCNA study guide. We have rather a large influence over quite a quantity of candidates. We are more than more popular by our high passing rate and high quality of our KCNA Study Guide. Our education team of professionals will give you the best of what you deserve. If you are headache about your KCNA certification exams, our KCNA training materials will be your best select.

The KCNA exam covers a wide range of topics related to Kubernetes and cloud-native technologies, including containerization, microservices, deployment, and management. It is designed to be accessible to individuals with a variety of backgrounds and skill levels, from beginners who are just starting to learn about these technologies to experienced professionals who have been working with them for years. KCNA Exam is also designed to be flexible, allowing individuals to take it at their own pace and on their own schedule, using a variety of study materials and resources.

Linux Foundation Kubernetes and Cloud Native Associate Sample Questions (Q111-Q116):

NEW QUESTION # 111
What's the difference between a security profile and a security context?

- A. Security Profiles configure Pods and Containers at runtime. Security Contexts are control plane mechanisms to enforce specific settings in the Security Profile.
- B. Security Contexts configure Pods and Containers at runtime. Security profiles are control plane mechanisms to enforce specific settings in the Security Context.
- C. Security Contexts configure Clusters and Namespaces at runtime. Security profiles are control plane mechanisms to enforce specific settings in the Security Context.
- D. Security Profiles configure Clusters and Namespaces at runtime. Security Contexts are control plane mechanisms to enforce specific settings in the Security Profile.

Answer: B

Explanation:
The correct answer is B. In Kubernetes, a securityContext is part of the Pod and container specification that configures runtime security settings for that workload-things like runAsUser, runAsNonRoot, Linux capabilities, readOnlyRootFilesystem, allowPrivilegeEscalation, SELinux options, seccomp profile selection, and filesystem group (fsGroup). These settings directly affect how the Pod's containers run on the node.
A security profile, in contrast, is a higher-level policy/standard enforced by the cluster control plane (typically via admission control) to ensure workloads meet required security constraints. In modern Kubernetes, this concept aligns with mechanisms like Pod Security Standards (Privileged, Baseline, Restricted) enforced through Pod Security Admission. The "profile" defines what is allowed or forbidden (for example, disallow privileged containers, disallow hostPath mounts, require non-root, restrict capabilities). The control plane enforces these constraints by validating or rejecting Pod specs that do not comply-ensuring consistent security posture across namespaces and teams.
Option A and D are incorrect because security contexts do not "configure clusters and namespaces at runtime"; security contexts apply to Pods/containers. Option C reverses the relationship: security profiles don't configure Pods at runtime; they constrain what security context settings (and other fields) are acceptable.
Practically, you can think of it as:
SecurityContext = workload-level configuration knobs (declared in manifests, applied at runtime).
SecurityProfile/Standards = cluster-level guardrails that determine which knobs/settings are permitted.
This separation supports least privilege: developers declare needed runtime settings, and cluster governance ensures those settings stay within approved boundaries. Therefore, B is the verified answer.

NEW QUESTION # 112
What is the main purpose of the Ingress in Kubernetes?

- A. Access services different from HTTP or HTTPS running in the cluster based on their IP address.
- B. Access HTTP and HTTPS services running in the cluster based on their path.
- C. Access HTTP and HTTPS services running in the cluster based on their IP address.
- D. Access services different from HTTP or HTTPS running in the cluster based on their path.

Answer: B

Explanation:
D is correct. Ingress is a Kubernetes API object that defines rules for external access to HTTP/HTTPS services in a cluster. The defining capability is Layer 7 routing-commonly host-based and path-based routing-so you can route requests like example.com/app1 to one Service and example.com/app2 to another.
While the question mentions "based on their path," that's a classic and correct Ingress use case (and host routing is also common). Ingress itself is only the specification of routing rules. An Ingress controller (e.g., NGINX Ingress Controller, HAProxy, Traefik, cloud-provider controllers) is what actually implements those rules by configuring a reverse proxy/load balancer. Ingress typically

terminates TLS (HTTPS) and forwards traffic to internal Services, giving a more expressive alternative to exposing every service via NodePort/LoadBalancer.
Why the other options are wrong:
* A suggests routing by IP address; Ingress is fundamentally about HTTP(S) routing rules (host/path), not direct Service IP access.
* B and C describe non-HTTP protocols; Ingress is specifically for HTTP/HTTPS. For TCP/UDP or other protocols, you generally use Services of type LoadBalancer/NodePort, Gateway API implementations, or controller-specific TCP/UDP configuration.
Ingress is a foundational building block for cloud-native application delivery because it centralizes edge routing, enables TLS management, and supports gradual adoption patterns (multiple services under one domain). Therefore, the main purpose described here matches D.
==========

NEW QUESTION # 113
What is container runtime?

- A. Software that runs containers
- B. Another term of kubelet or kubectl
- C. The amount of time it takes a container to execute
- D. A container image format

Answer: A

Explanation:
https://www.aquasec.com/cloud-native-academy/container-security/container-runtime/

NEW QUESTION # 114
Which of the following options include resources cleaned by the Kubernetes garbage collection mechanism?

- A. Stale or expired CertificateSigningRequests (CSRs) and old deployments.
- B. Unused container and container images, and obsolete logs from the kubelet.
- C. Nodes deleted by a cloud controller manager and obsolete logs from the kubelet.
- D. Terminated pods, completed jobs, and objects without owner references.

Answer: D

Explanation:
Kubernetes garbage collection (GC) is about cleaning up API objects and related resources that are no longer needed, so the correct answer is D. Two big categories it targets are (1) objects that have finished their lifecycle (like terminated Pods and completed Jobs, depending on controllers and TTL policies), and (2) "dangling" objects that are no longer referenced properly-often described as objects without owner references (or where owners are gone), which can happen when a higher-level controller is deleted or when dependent resources are left behind.
A key Kubernetes concept here is OwnerReferences: many resources are created "owned" by a controller (e.g., a ReplicaSet owned by a Deployment, Pods owned by a ReplicaSet). When an owning object is deleted, Kubernetes' garbage collector can remove dependent objects based on deletion propagation policies (foreground/background/orphan). This prevents resource leaks and keeps the cluster tidy and performant.
The other options are incorrect because they refer to cleanup tasks outside Kubernetes GC's scope. Kubelet logs (B/C) are node-level files and log rotation is handled by node/runtime configuration, not the Kubernetes garbage collector. Unused container images (C) are managed by the container runtime's image GC and kubelet disk pressure management, not the Kubernetes API GC. Nodes deleted by a cloud controller (B) aren't "garbage collected" in the same sense; node lifecycle is handled by controllers and cloud integrations, but not as a generic GC cleanup category like ownerRef-based object deletion.
So, when the question asks specifically about "resources cleaned by Kubernetes garbage collection," it's pointing to Kubernetes object lifecycle cleanup: terminated Pods, completed Jobs, and orphaned objects-exactly what option D states.

NEW QUESTION # 115
Which of the following sentences is true about namespaces in Kubernetes?

- A. You can create two resources of the same kind and name in a namespace.
- B. All the objects in the cluster are namespaced by default.

- C. You can create a namespace within another namespace in Kubernetes.
- D. The default namespace exists when a new cluster is created.

Answer: D

Explanation:
The true statement is C: the default namespace exists when a new cluster is created. Namespaces are a Kubernetes mechanism for partitioning cluster resources into logical groups. When you set up a cluster, Kubernetes creates some initial namespaces (including default, and commonly kube-system, kube-public, and kube-node-lease). The default namespace is where resources go if you don't specify a namespace explicitly.
Option A is false because namespaces are not hierarchical; Kubernetes does not support "namespaces inside namespaces." Option B is false because within a given namespace, resource names must be unique per resource kind. You can't have two Deployments with the same name in the same namespace. You can have a Deployment named web in one namespace and another Deployment named web in a different namespace-namespaces provide that scope boundary. Option D is false because not all objects are namespaced. Many resources are cluster-scoped (for example, Nodes, PersistentVolumes, ClusterRoles, ClusterRoleBindings, and StorageClasses). Namespaces apply only to namespaced resources.
Operationally, namespaces support multi-tenancy and environment separation (dev/test/prod), RBAC scoping, resource quotas, and policy boundaries. For example, you can grant a team access only to their namespace and enforce quotas that prevent them from consuming excessive CPU/memory. Namespaces also make organization and cleanup easier: deleting a namespace removes most namespaced resources inside it (subject to finalizers).
So, the verified correct statement is C: the default namespace exists upon cluster creation.

NEW QUESTION # 116
......

myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, Disposable vapes

BONUS!!! Download part of Prep4SureReview KCNA dumps for free: https://drive.google.com/open?id=1ROyKVbRu-GeP8EZaqeIMG4FMhE4_tumr