

Pass Guaranteed Quiz 2026 KCNA: Kubernetes and Cloud Native Associate High Hit-Rate New Test Review



BTW, DOWNLOAD part of BraindumpsIT KCNA dumps from Cloud Storage: <https://drive.google.com/open?id=1IPhY483LafG-oSjTMuQMZPzmkAOFaw->

BraindumpsIT is a wonderful study platform that can transform your effective diligence in to your best rewards. By years of diligent work, our experts have collected the frequent-tested knowledge into our KCNA exam materials for your reference. So our KCNA Practice Questions are triumph of their endeavor. I can say that no one can know the KCNA study guide better than them and our quality of the KCNA learning quiz is the best.

The KCNA Exam is a performance-based exam that involves completing a set of tasks in a real-world environment. KCNA exam is conducted online and requires candidates to have access to a Linux environment with Kubernetes and other cloud-native tools installed. KCNA Exam is designed to test the candidate's ability to deploy and manage containerized applications using Kubernetes and other cloud-native technologies.

>> New KCNA Test Review <<

KCNA Relevant Questions | KCNA Exam Reference

Obtaining this KCNA certificate is not an easy task, especially for those who are busy every day. However, if you use our KCNA exam torrent, we will provide you with a comprehensive service to overcome your difficulties and effectively improve your ability. If you can take the time to learn about our KCNA Quiz prep, I believe you will be interested in our KCNA exam questions. Our KCNA learning materials are practically tested, choosing our KCNA exam guide, you will get unexpected surprise.

Linux Foundation KCNA (Kubernetes and Cloud Native Associate) Exam is a certification program that tests an individual's knowledge of Kubernetes and cloud-native technologies. KCNA exam is designed to validate the skills and expertise required for working with Kubernetes and cloud-native environments. Kubernetes and Cloud Native Associate certification program is developed by the Linux Foundation, a non-profit organization that focuses on the promotion and development of open-source software.

Linux Foundation Kubernetes and Cloud Native Associate Sample Questions (Q40-Q45):

NEW QUESTION # 40

What is the command used to login to the pod?

- A. kubectl login

- B. kubectl list
- C. kubectl exec
- D. kubectl get

Answer: C

Explanation:

<https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#exec>

List contents of /usr from the first container of pod mypod and sort by modification time # If the command you want to execute in the pod has any flags in common (e.g. -i), # you must use two dashes (--) to separate your command's flags/arguments # Also note, do not surround your command and its flags/arguments with quotes # unless that is how you would execute it normally (i.e., do ls -t /usr, not "ls -t /usr")

`kubectl exec mypod -i -t -- ls -t /usr`

NEW QUESTION # 41

Notary and the update framework leading security projects in CNCF

- A. TRUE
- B. FALSE

Answer: A

Explanation:

<https://github.com/cncf/landscape#trail-map>

CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape l.cncf.io has a large number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator or a Certified Kubernetes Application Developer l.cncf.io/training

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider l.cncf.io/kscsp

C. Join CNCF's End User Community

For companies that don't offer cloud native services externally l.cncf.io/enduser

WHAT IS CLOUD NATIVE?

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible for everyone.

l.cncf.io

v20200501



1. CONTAINERIZATION

- Commonly done with Docker containers
- Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized.
- Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices

3. ORCHESTRATION & APPLICATION DEFINITION

- Kubernetes is the market-leading orchestration solution
- You should select a Certified Kubernetes Distribution, Hosted Platform, or Installer: [cncf.io/ck](https://l.cncf.io/ck)
- Helm Charts help you define, install, and upgrade even the most complex Kubernetes application



5. SERVICE PROXY, DISCOVERY, & MESH

- CoreDNS is a fast and flexible tool that is useful for service discovery
- Envoy and Linkerd each enable service mesh architectures
- They offer health checking, routing, and load balancing



7. DISTRIBUTED DATABASE & STORAGE

When you need more resiliency and scalability than you can get from a single database, Vitess is a good option for running MySQL at scale through sharding. Rook is a storage orchestrator that integrates a diverse set of storage solutions into Kubernetes. Serving as the "brain" of Kubernetes, etcd provides a reliable way to store data across a cluster of machines. TiKV is a high performance distributed transactional key-value store written in Rust.



9. CONTAINER REGISTRY & RUNTIME

Harbor is a registry that stores, signs, and scans content. You can use alternative container runtimes. The most common, both of which are OCI-compliant, are containerd and CRI-O.



2. CI/CD

- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
- Setup automated rollouts, roll backs and testing
- Argo is a set of Kubernetes-native tools for deploying and running jobs, applications, workflows, and events using GitOps paradigms such as continuous and progressive delivery and MLOps



4. OBSERVABILITY & ANALYSIS

- Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus for monitoring, Fluentd for logging and Jaeger for Tracing
- For tracing, look for an OpenTracing-compatible implementation like Jaeger



6. NETWORKING, POLICY, & SECURITY

To enable more flexible networking, use a CNI-compliant network project like Calico, Flannel, or Weave Net. Open Policy Agent (OPA) is a general purpose policy engine with uses ranging from authorization and admission control to data filtering. Falco is an anomaly detection engine for cloud native.



8. STREAMING & MESSAGING

When you need higher performance than JSON-REST, consider using gRPC or NATS. gRPC is a universal RPC framework. NATS is a multi-modal messaging system that includes request/reply, pub/sub and load balanced queues. CloudEvents is a specification for describing event data in common ways.



10. SOFTWARE DISTRIBUTION

If you need to do secure software distribution, evaluate Notary, an implementation of The Update Framework.



NEW QUESTION # 42

Consider a pod with a "readinessProbe" that checks for a specific file existence. Explain what happens if the probe fails repeatedly, and how it affects the pod's lifecycle.

- A. The pod will be automatically scaled down.
- B. The pod will be automatically restarted.
- **C. The pod will be marked as "unhealthy", but will continue to run.**
- D. The pod will be rescheduled to a different node in the cluster.
- E. The pod will be terminated and removed from the cluster.

Answer: C

Explanation:

If the "readinessProbe" fails repeatedly, the pod will be marked as "unhealthy", but it will continue to run. The "readinessProbe" is responsible for ensuring that the pod is ready to receive traffic. If the probe fails, it indicates that the pod is not yet ready, and Kubernetes will not direct traffic to it. The pod will not be restarted, terminated, rescheduled, or scaled down. It will remain in an unhealthy state, and traffic will only be directed to it once the probe starts succeeding. This allows the pod to continue running while it resolves the issue causing the probe failures.

NEW QUESTION # 43

Which resource do you use to attach a volume in a Pod?

- A. StorageVolume
- B. PersistentVolumeClaim
- C. PersistentVolume
- D. StorageClass

Answer: B

Explanation:

In Kubernetes, Pods typically attach persistent storage by referencing a PersistentVolumeClaim (PVC), making D correct. A PVC is a user's request for storage with specific requirements (size, access mode, storage class). Kubernetes then binds the PVC to a matching PersistentVolume (PV) (either pre-provisioned statically or created dynamically via a StorageClass and CSI provisioner). The Pod does not directly attach a PV; it references the PVC, and Kubernetes handles the binding and mounting.

This design separates responsibilities: administrators (or CSI drivers) manage PV provisioning and backend storage details, while developers consume storage via PVCs. In a Pod spec, you define a volume of type persistentVolumeClaim and set claimName: <pvc-name>, then mount that volume into containers at a path. The kubelet coordinates with the CSI driver (or in-tree plugin depending on environment) to attach/mount the underlying storage to the node and then into the Pod.

Option B (PersistentVolume) is not directly referenced by Pods; PVs are cluster resources that represent actual storage. Pods don't "pick" PVs; claims do. Option C (StorageClass) defines provisioning parameters (e.g., disk type, replication, binding mode) but is not what a Pod references to mount a volume. Option A is not a Kubernetes resource type.

Operationally, using PVCs enables dynamic provisioning and portability: the same Pod spec can be deployed across clusters where the StorageClass name maps to appropriate backend storage. It also supports lifecycle controls like reclaim policies (Delete/Retain) and snapshot/restore workflows depending on CSI capabilities.

So the Kubernetes resource you use in a Pod to attach a persistent volume is PersistentVolumeClaim, option D.

NEW QUESTION # 44

What is a key feature of a container network?

- A. Allowing containers running on separate hosts to communicate.
- B. Proxying REST requests across a set of containers.
- C. Allowing containers on the same host to communicate.
- D. Caching remote disk access.

Answer: A

Explanation:

A defining requirement of container networking in orchestrated environments is enabling workloads to communicate across hosts, not just within a single machine. That's why B is correct: a key feature of a container network is allowing containers (Pods) running on separate hosts to communicate.

In Kubernetes, this idea becomes the Kubernetes network model: every Pod gets an IP address, and Pods should be able to communicate with other Pods across nodes without needing NAT (depending on implementation details). Achieving that across a cluster requires a networking layer (typically implemented by a CNI plugin) that can route traffic between nodes so that Pod-to-Pod communication works regardless of placement. This is crucial because schedulers dynamically place Pods; you cannot assume two communicating components will land on the same node.

Option C is true in a trivial sense—containers on the same host can communicate—but that capability alone is not the key feature that makes orchestration viable at scale. Cross-host connectivity is the harder and more essential property. Option A describes application-layer behavior (like API gateways or reverse proxies) rather than the foundational networking capability. Option D describes storage optimization, unrelated to container networking.

From a cloud native architecture perspective, reliable cross-host networking enables microservices patterns, service discovery, and distributed systems behavior. Kubernetes Services, DNS, and NetworkPolicies all depend on the underlying ability for Pods across the cluster to send traffic to each other. If your container network cannot provide cross-node routing and reachability, the cluster

