Kubernetes and Cloud Native Associate (KCNA)

Linux Foundation KCNA

Version Demo

Total Demo Questions: 9

Total Premium Questions: 126

QUESTION NO: 1

What is a benefits of Kubernetes federation?

A. Avoids scalability limits on pods and nodes

B. Creates highly available clusters in different regions

C. Low latency

ANSWER: A B C

QUESTION NO: 2

What kind of limitation cgroups allows?

A. Prioritization

B. Resource limiting

C. Accounting

D. None of the options

E. Control

F. Server cpu and memory

ANSWER: A B C E

QUESTION NO: 3

Which control plane component is responsible for scheduling pods?

A. kube-proxy

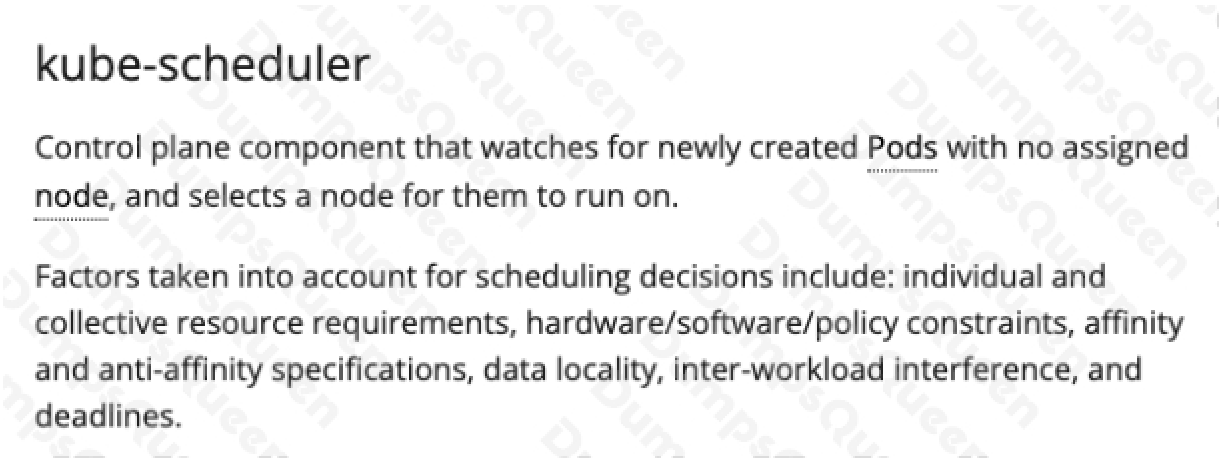
B. kube scheduler

C. kubelet

D. kube api-server

ANSWER: B

Explanation:



kube-scheduler

Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

Factors taken into account for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.

QUESTION NO: 4

What are container runtimes with Kubernetes?

A. CRI-O

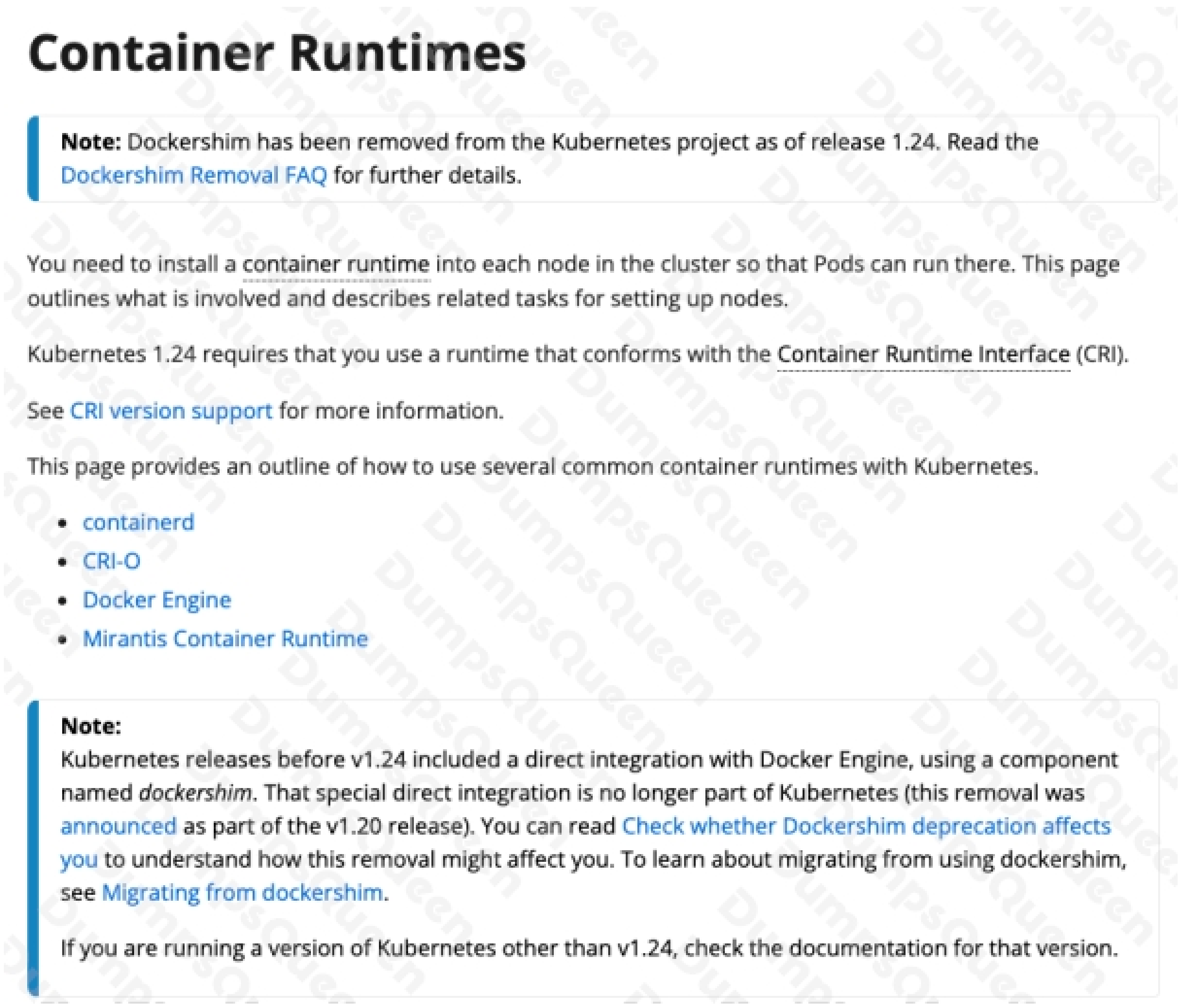
B. lxd

C. containerd

D. Dockershim

ANSWER: A C

Explanation:



Container Runtimes

Note: Dockershim has been removed from the Kubernetes project as of release 1.24. Read the Dockershim Removal FAQ for further details.

You need to install a container runtime into each node in the cluster so that Pods can run there. This page outlines what is involved and describes related tasks for setting up nodes.

Kubernetes 1.24 requires that you use a runtime that conforms with the Container Runtime Interface (CRI).

See CRI version support for more information.

This page provides an outline of how to use several common container runtimes with Kubernetes.

- containerd
- CRI-O
- Docker Engine
- Mirantis Container Runtime

Note:
Kubernetes releases before v1.24 included a direct integration with Docker Engine, using a component named *dockershim*. That special direct integration is no longer part of Kubernetes (this removal was announced as part of the v1.20 release). You can read Check whether Dockershim deprecation affects you to understand how this removal might affect you. To learn about migrating from using dockershim, see Migrating from dockershim.

If you are running a version of Kubernetes other than v1.24, check the documentation for that version.

QUESTION NO: 5

What do control groups provide when it come to containers

A. Permission

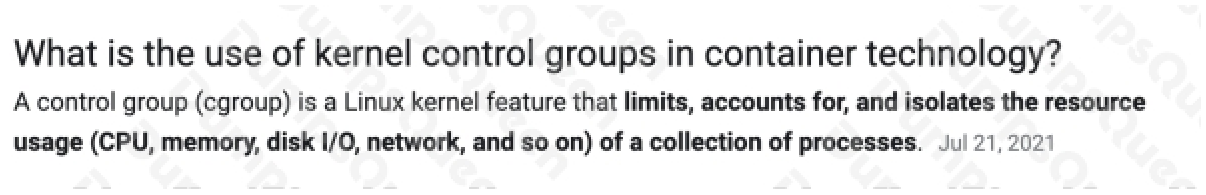
B. Image Storage

C. Isolation

D. Logging

ANSWER: C

Explanation:



What is the use of kernel control groups in container technology?

A control group (cgroup) is a Linux kernel feature that **limits, accounts for, and isolates the resource usage (CPU, memory, disk I/O, network, and so on) of a collection of processes**. Jul 21, 2021

QUESTION NO: 6

Various Container Orchestrator Systems (COS)?

A. Apache Mesos

B. None of the options

C. Docker Swarm

D. Kubernetes

ANSWER: A C D

QUESTION NO: 7

Observability and monitoring are not the same?

A. True

B. False

ANSWER: A

QUESTION NO: 8

Which statement is true about Pod Networking?

A. All pod requires an external DNS server to get the hostname

B. All containers in a pod get a unique IP address

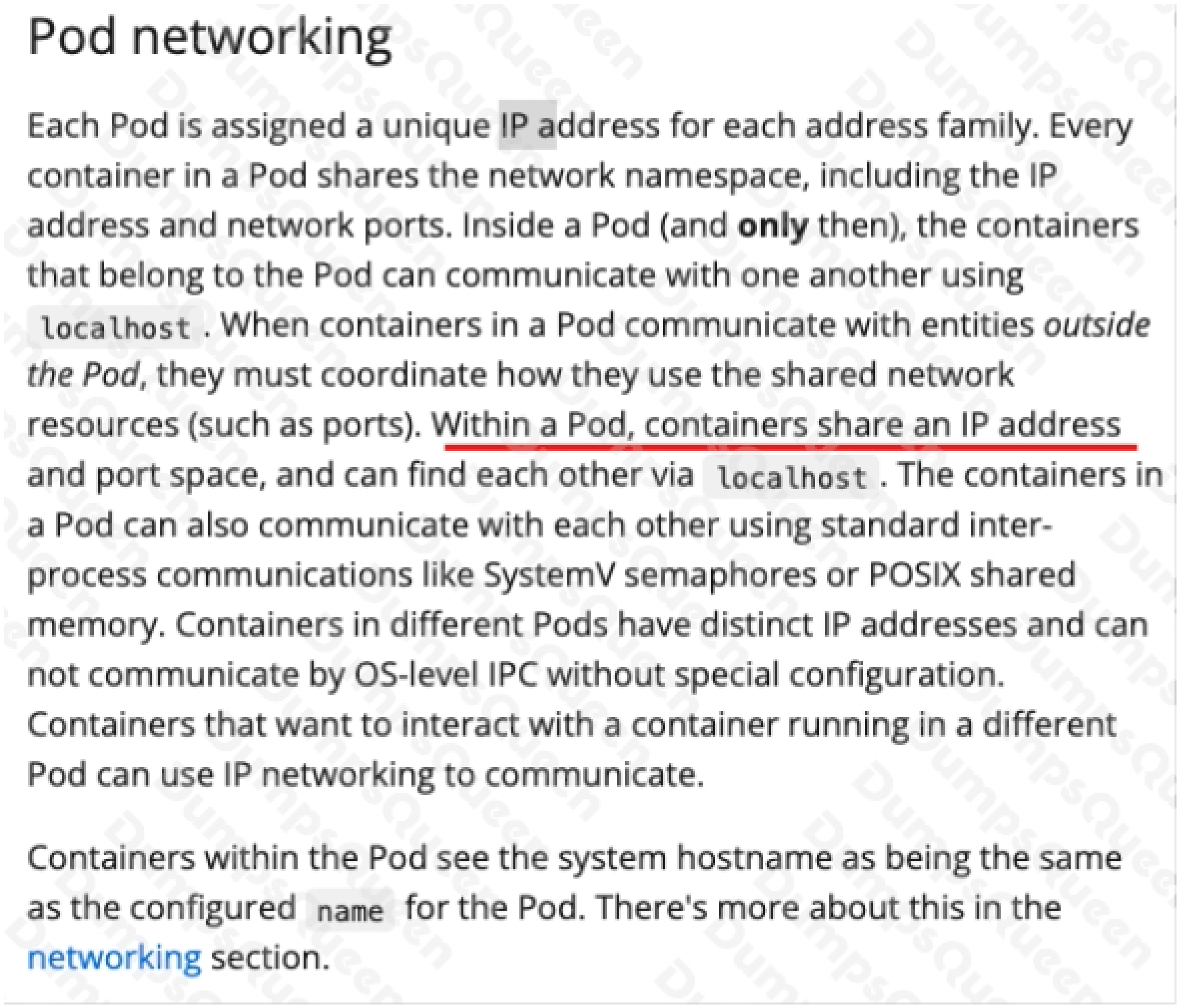
C. All containers in a pod share a single IP address

D. All pod requires NAT to get a unique IP address.

ANSWER: C

Explanation:

https://kubernetes.io/docs/concepts/workloads/pods/#pod-networking



Pod networking

Each Pod is assigned a unique IP address for each address family. Every container in a Pod shares the network namespace, including the IP address and network ports. Inside a Pod (and **only** then), the containers that belong to the Pod can communicate with one another using `localhost`. When containers in a Pod communicate with entities *outside the Pod*, they must coordinate how they use the shared network resources (such as ports). Within a Pod, containers share an IP address and port space, and can find each other via `localhost`. The containers in a Pod can also communicate with each other using standard inter-process communications like SystemV semaphores or POSIX shared memory. Containers in different Pods have distinct IP addresses and can not communicate by OS-level IPC without special configuration. Containers that want to interact with a container running in a different Pod can use IP networking to communicate.

Containers within the Pod see the system hostname as being the same as the configured `name` for the Pod. There's more about this in the networking section.

QUESTION NO: 9

What standard does kubelet use to communicate with the container runtime?

A. Service Mesh Interface (SMI)

B. CRI-O

C. ContainerD

D. Container Runtime Interface (CRI)

ANSWER: D