KCNA Q&As

Kubernetes and Cloud Native Associate (KCNA)

Pass Linux Foundation KCNA Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.pass4itsure.com/kcna.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Linux Foundation Official Exam Center

QUESTION 1

What is the functionality of the daemon set?

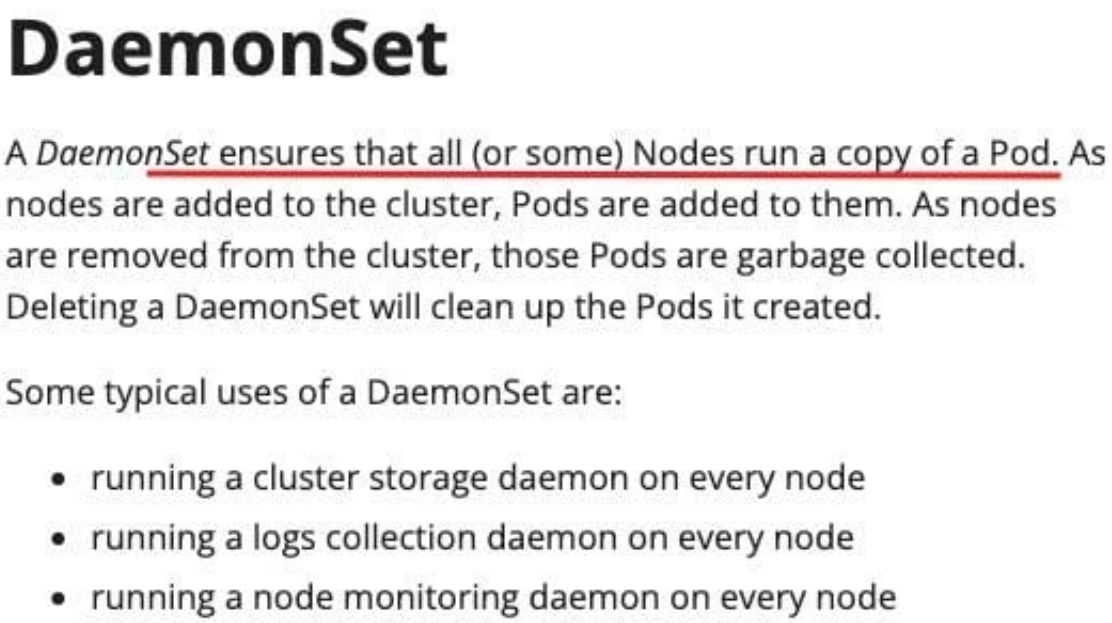
A. To run a copy of the pod in all the nodes of the cluster

B. To initialize the pod before starting the main pod

C. To run a copy of the pod in a single node of the cluster

Correct Answer: A

Explanation: https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/



QUESTION 2

Which of the following best describes a cloud-native app?

A. An application where all logic is coded into a single large binary.

B. An application that publishes an HTTPS web front-end.

C. An application that takes advantages of cloud computing fromworks and their loosely coupled cloud services.

D. An application that leverages services that are native to public cloud platforms such as Azure, GCP, and/or AWS.

Correct Answer: C

Explanation: Cloud-native apps leverage cloud computing frameworks and tend to be microservices based, where individual components of the app are coded as individual.

QUESTION 3

What is the primary interface for Kubernetes cluster?

A. Kubernetes Api

B. Kubelet

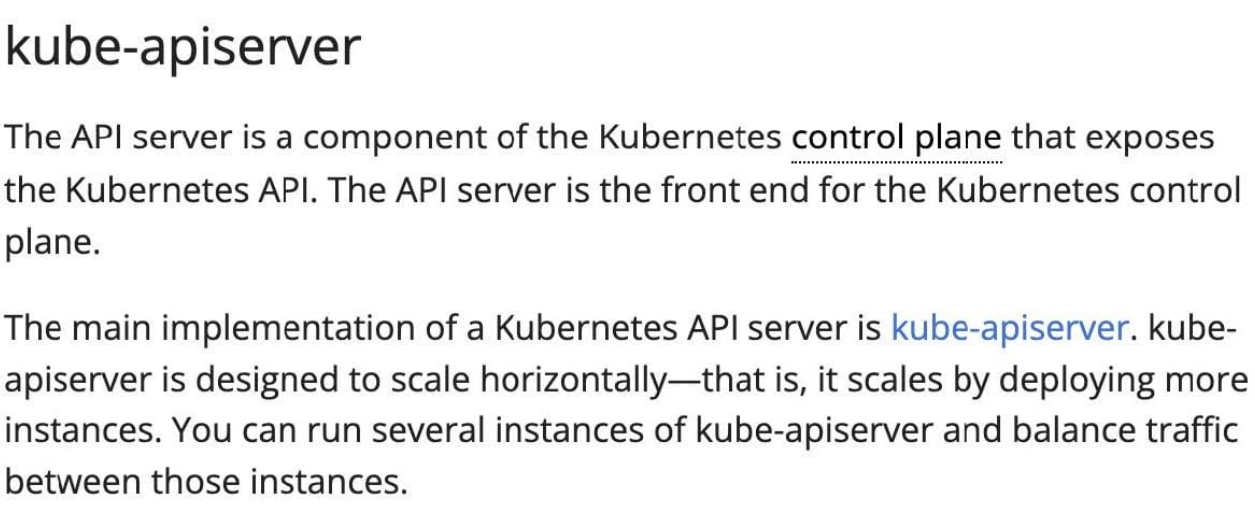
C. YAML

D. Control Plane

E. JSON

Correct Answer: A

Explanation: https://kubernetes.io/docs/concepts/overview/components/#kube-apiserver



QUESTION 4

Which statement is true about Pod Networking?

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

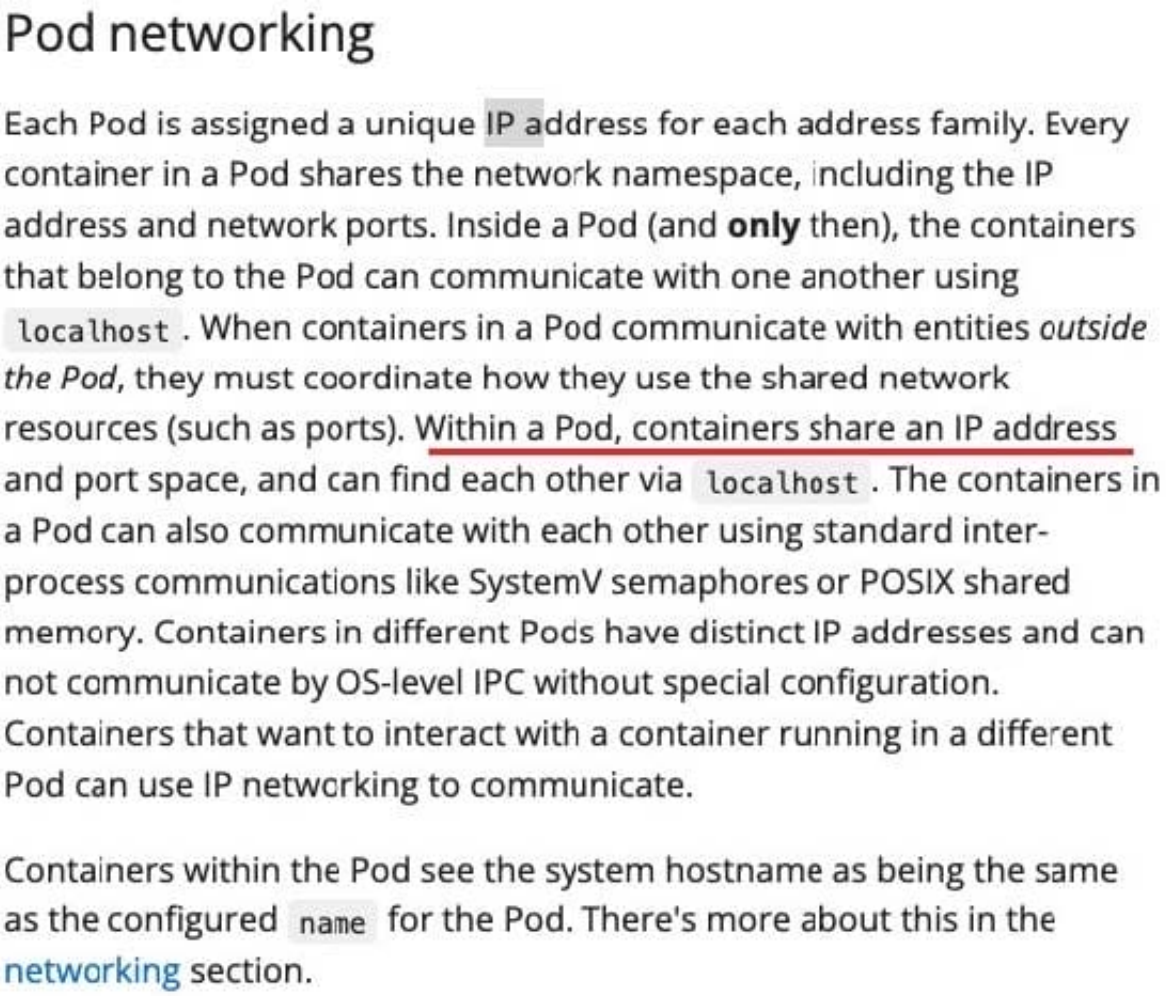
B. All containers in a pod get a unique IP address

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

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

Correct 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 5

What is the default service type in Kubernetes?

A. CusterIP

B. NodePort

C. serviceType

D. loadBalancer

Correct Answer: A

Explanation: https://kubernetes.io/docs/concepts/services-networking/service/#publishing- services-service-types

Kubernetes `ServiceTypes` allow you to specify what kind of Service you want. The default is `ClusterIP`.

`Type` values and their behaviors are:

- `ClusterIP` : Exposes the Service on a cluster-internal IP. Choosing this value makes the Service only reachable from within the cluster. This is the default `ServiceType`.
- `NodePort`: Exposes the Service on each Node's IP at a static port (the `NodePort`). A `ClusterIP` Service, to which the `NodePort` Service routes, is automatically created. You'll be able to contact the `NodePort` Service, from outside the cluster, by requesting `<NodeIP>:<NodePort>`.
- `LoadBalancer`: Exposes the Service externally using a cloud provider's load balancer. `NodePort` and `ClusterIP` Services, to which the external load balancer routes, are automatically created.
- `ExternalName`: Maps the Service to the contents of the `externalName` field (e.g. `foo.bar.example.com`), by returning a `CNAME` record with its value. No proxying of any kind is set up.