



# KCNA<sup>Q&As</sup>

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.geekcert.com/kcna.html>

100% Passing Guarantee  
100% Money Back Assurance

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

- ⚙️ **Instant Download** After Purchase
- ⚙️ **100% Money Back** Guarantee
- ⚙️ **365 Days** Free Update
- ⚙️ **800,000+** Satisfied Customers





### QUESTION 1

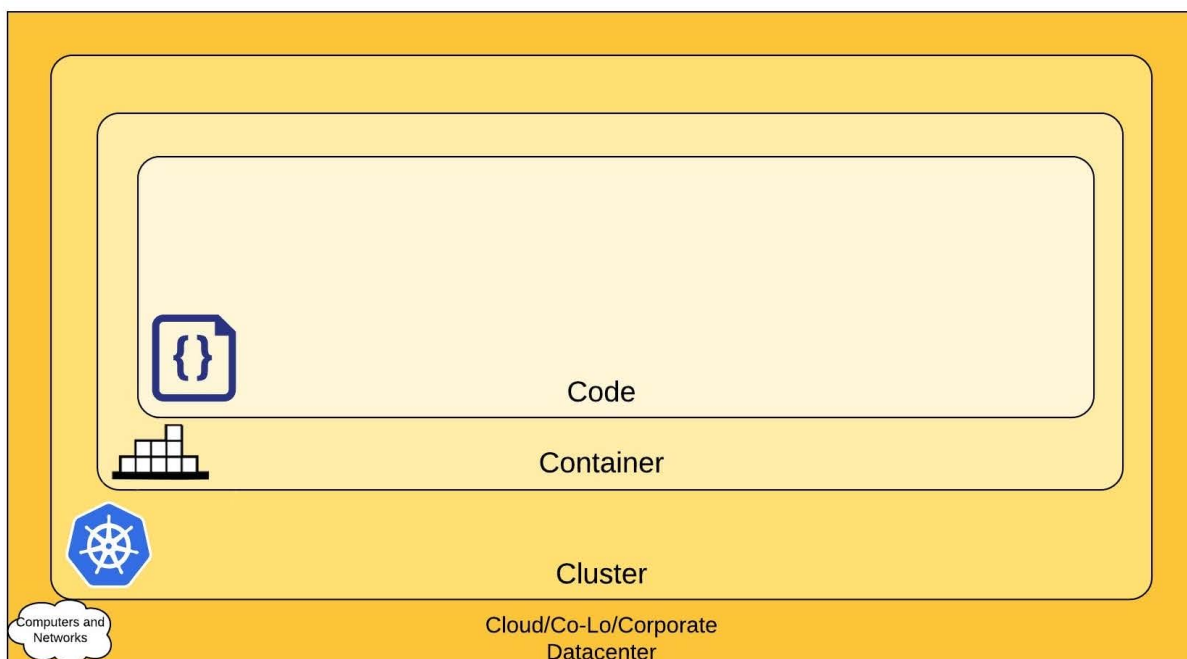
There are three Nodes in a cluster, and want to run exactly one replica of a Pod on each Node. Prefer to automatically create a replica on any new Nodes when they are added.

Which Kubernetes re-source should you use?

- A. DaemonSet
- B. ReplicaSet
- C. NodeSet
- D. StatefulSet
- E. Deployment

Correct Answer: A

<https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/> A DaemonSet runs replicas on all (or just some) Nodes in the cluster.





## QUESTION 2

The Kubernetes rolling update is used for \_\_\_\_.

- A. Updating a service
- B. Scaling an application
- C. Updating a deployment

Correct Answer: C

Explanation: <https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/>



# Performing a Rolling Update

## Objectives

- Perform a rolling update using kubectl.

## Updating an application

Users expect applications to be available all the time and developers are expected to deploy new versions of them several times a day. In Kubernetes this is done with rolling updates. **Rolling updates** allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones. The new Pods will be scheduled on Nodes with available resources.

In the previous module we scaled our application to run multiple instances. This is a requirement for performing updates without affecting application availability. By default, the maximum number of Pods that can be unavailable during the update and the maximum number of new Pods that can be created, is one. Both options can be configured to either numbers or percentages (of Pods). In Kubernetes, updates are versioned and any Deployment update can be reverted to a previous (stable) version.

### Summary:

- Updating an app

*Rolling updates allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones.*

## QUESTION 3

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 4

What is the name for a service that has no clusterIp address?

- A. Headless
- B. NodePort
- C. ClusterIP
- D. LoadBalancer

Correct Answer: A

Explanation: <https://kubernetes.io/docs/concepts/services-networking/service/#headless-services>



# Headless Services

Sometimes you don't need load-balancing and a single Service IP. In this case, you can create what are termed "headless" Services, by explicitly specifying "None" for the cluster IP ( `.spec.clusterIP` ).

You can use a headless Service to interface with other service discovery mechanisms, without being tied to Kubernetes' implementation.

For headless Services , a cluster IP is not allocated, kube-proxy does not handle these Services, and there is no load balancing or proxying done by the platform for them. How DNS is automatically configured depends on whether the Service has selectors defined:

---

## QUESTION 5

What is horizontal scaling?

- A. Creating a Deployment
- B. Adding resources to existing apps and servers
- C. Moving workloads from one server to another
- D. Adding additional replicas of apps and servers

Correct Answer: D

Explanation: <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>



In Kubernetes, a *HorizontalPodAutoscaler* automatically updates a workload resource (such as a Deployment or StatefulSet), with the aim of automatically scaling the workload to match demand.

Horizontal scaling means that the response to increased load is to deploy more Pods. This is different from *vertical* scaling, which for Kubernetes would mean assigning more resources (for example: memory or CPU) to the Pods that are already running for the workload.

If the load decreases, and the number of Pods is above the configured minimum, the *HorizontalPodAutoscaler* instructs the workload resource (the Deployment, StatefulSet, or other similar resource) to scale back down.

Horizontal pod autoscaling does not apply to objects that can't be scaled (for example: a DaemonSet.)

The *HorizontalPodAutoscaler* is implemented as a Kubernetes API resource and a controller. The resource determines the behavior of the controller. The horizontal pod autoscaling controller, running within the Kubernetes control plane, periodically adjusts the desired scale of its target (for example, a Deployment) to match observed metrics such as average CPU utilization, average memory utilization, or any other custom metric you specify.

[Latest KCNA Dumps](#)

[KCNA Study Guide](#)

[KCNA Exam Questions](#)