



CERTSWARRIOR

Linux Foundation KCNA

Kubernetes and Cloud Native Associate

Questions&AnswersPDF

ForMoreInformation:

<https://www.certswarrior.com/>

Features:

- 90DaysFreeUpdates
- 30DaysMoneyBackGuarantee
- InstantDownloadOncePurchased
- 24/7OnlineChat Support
- ItsLatestVersion

Latest Version: 6.0

Question: 1

Consider the following Kubernetes YAML configuration for a Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```

What is the purpose of the 'replicas' field in this configuration, and how does it affect the Deployment? The 'replicas' field specifies the number of Pods that should be created and maintained by the Deployment. It controls the desired number of running instances of the application.

- A. The 'replicas' field defines the minimum number of Pods that should be available at all times. It ensures that the application remains available even if some Pods fail.
- B. The 'replicas' field specifies the maximum number of Pods that can be created by the Deployment. It limits the amount of resources that the application can consume.
- C. The 'replicas' field specifies the initial number of Pods that should be created when the Deployment is deployed. It does not affect the number of Pods that are running after the initial deployment.
- D. The 'replicas' field specifies the number of nodes in the Kubernetes cluster. It determines the number of Pods that can be scheduled

Answer: A

Explanation:

The 'replicas' field in a Deployment specifies the number of Pods that should be created and maintained by the Deployment. It controls the desired number of running instances of the application. If a Pod fails, the Deployment will automatically create a new Pod to replace it, ensuring that the desired number of replicas is maintained.

Question: 2

How does Kubernetes ensure the high availability of a Deployment when running in a multi-node cluster?

- A. By using the 'replicas' field in the Deployment configuration, Kubernetes ensures that multiple instances of the application are running across different nodes. If one node fails, the Pods running on that node are automatically restarted on other healthy nodes.
- B. Kubernetes uses a distributed consensus algorithm called etcd to store the state of the cluster. This ensures that even if one node fails, the other nodes have access to the latest state information and can continue to operate-
- C. Kubernetes automatically scales up the number of Pods running in a Deployment if a node fails- This ensures that there are enough Pods to handle the workload
- D. Kubernetes uses a technique called rolling updates to ensure that the application remains available during deployments. This means that new Pods are deployed and brought online gradually, while old Pods are gracefully shut down.
- E. Kubernetes uses a combination of the 'replicas' field, the etcd distributed database, and the self-healing capabilities of the platform to ensure the high availability of Deployments in a multi-node cluster.

Answer: E

Explanation:

Kubernetes achieves high availability for Deployments through a combination of techniques: Replicas: The 'replicas' field ensures that multiple instances of the application are running on different nodes. If one node fails, the Pods running on that node are automatically restarted on other healthy nodes. etcd: The distributed database etcd stores the state of the cluster. This means that even if one node fails, the other nodes have access to the latest state information and can continue to operate. Self-healing: Kubernetes constantly monitors the health of Pods and restarts failed Pods on healthy nodes. This ensures that the desired number of replicas is always maintained, even if some Pods experience failures. These elements work together to ensure that applications remain available even in the face of node failures.

Question: 3

You have a Kubernetes cluster with a deployment named 'my-app' that has a replica count of 3. You need to temporarily scale it down to 1 replica. Which command would you use?

- A. `kubectl scale deployments my-app --replicas=1`
- B. `kubectl edit deployment my-app --replicas=1`
- C. `kubectl patch deployments my-app --type=json--patch='{"spec":{"replicas":1}}'`
- D. `kubectl set scale deployments my-app --replicas=1`

E. `kubectl apply -f my-app-deployment.yaml --replicas=1`

Answer: A,D

Explanation:

Both '`kubectl scale deployments my-app --replicas=1`' and '`kubectl set scale deployments my-app --replicas=1`' are valid commands to scale down the deployment. Option 'C' also works but it is more verbose. Option 'B' would edit the deployment configuration, which is not the desired outcome in this case. Option 'E' would apply a new configuration, which is also not ideal.

Question: 4

You are setting up a new Kubernetes cluster and need to configure a network policy that allows traff...
This content requires a subscription.

A.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-in-namespace
spec:
  podSelector: {}
  ingress:
    - from:
      - podSelector: {}
  egress: []
```

B.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-in-namespace
spec:
  podSelector: {}
  ingress:
    - from:
      - namespaceSelector: {}
  egress: []
```

C.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-in-namespace
spec:
  podSelector: {}
  ingress:
    - from:
      - ipBlock:
          cidr: 10.244.0.0/16
  egress: []
```

D.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-in-namespace
spec:
  podSelector: {}
  ingress:
    - from:
      - ipBlock:
          cidr: 172.17.0.0/16
  egress: []
```

E.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-in-namespace
spec:
  podSelector: {}
  ingress:
    - from:
      - ipBlock:
          cidr: 192.168.0.0/16
  egress: []
```

Answer: A

Explanation:

Option 'A' is the correct choice- It defines a network policy that allows traffic from Pods within the same namespace by using an empty •podSelectof _ Options 'B', 'D' and 'E' all define policies that either allow traffic from specific namespaces or from specific IP ranges, which are not suitable for this scenario.

Question: 5

Which Kubernetes API resource is responsible for defining the structure of your Kubernetes cluster, including the number and types of nodes?

- A. Deployment
- B. Service
- C. pod
- D. Node
- E. Cluster

Answer: E

Explanation:

The Cluster resource is responsible for defining the structure of the Kubernetes cluster, including the number and types of nodes. It is a top-level resource in Kubernetes. Deployments, Services, Pods, and Nodes are all lower-level resources that are defined within a cluster.

Question: 6

You have a Kubernetes deployment running on a cluster that includes multiple nodes. You need to ensure that Pods from this deployment are scheduled only on nodes with a specific label 'gpu=true'. How can you achieve this?

- A. By applying a NodeAffinity to the deployments pod template.
- B. By using a PodDisruptionBudget to ensure that only nodes with the 'gpu=true' label are considered for scheduling.
- C. By creating a custom resource definition that defines the GPU constraint and applying it to the deployment.
- D. By using a daemonset to run the deployment only on nodes with the 'gpu=true' label.
- E. By using a ReplicaSet to control the deployments replicas on nodes with the 'gpu=true' label.

Answer: A

Explanation:

Applying a NodeAffinity to the deployment's pod template is the correct way to schedule Pods on nodes with a specific label. NodeAffinity allows you to specify preferences or requirements for node selection based on labels, taints, and other criteria. Option 'B' is incorrect as PodDisruptionBudget deals with graceful pod termination during node maintenance. Options 'C' and 'D' are not the appropriate approach in this situation. Option 'E' is also incorrect because a ReplicaSet is used for managing replicas, not for node scheduling.

Question: 7

What is the purpose of the 'kubelet' component in Kubernetes?

- A. It manages the communication between the Kubernetes master and the nodes.
- B. It schedules Pods onto nodes based on resource availability and constraints.
- C. It runs and manages containers on each node, ensuring their health and resource usage.
- D. It manages the API server and provides access to Kubernetes resources.
- E. It is responsible for storing and managing Kubernetes configurations and state.

Answer: C

Explanation:

The 'kubelet' component in Kubernetes is responsible for running and managing containers on each node. It is the agent that communicates with the master and ensures that Pods are running as intended. Option 'A' is incorrect because that is the role of the kube-proxy and the API server. Option 'B' is incorrect because scheduling is handled by the scheduler. Option 'D' is incorrect because that is the role of the API server. Option 'E' is incorrect because configuration and state are managed by the etcd.

Question: 8

You need to create a Kubernetes service that exposes a TCP-based application on port 8080. You want the service to be accessible from external clients. Which type of service should you create?

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

Answer: B

Explanation:

The 'LoadBalancer' service type is the most suitable for exposing your TCP-based application on port 8080 to external clients. It will automatically create a load balancer in the cloud provider's infrastructure, allowing external access to your application. Option 'A' (ClusterIP) only allows access from within the cluster. Option 'C' (NodePort) exposes the service on a specific port on each node, making it accessible via the node's IP address. Option 'D' (ExternalName) is for exposing services that are already externally accessible using a DNS name. Option 'E' (Headless) is for services where you want to access Pods directly by their names, which is not the case here.

Question: 9

You have a Kubernetes deployment that runs an application with multiple replicas. When a Pod fails, you need to ensure that the deployment automatically creates a replacement Pod. Which Kubernetes API resource is responsible for this functionality?

- A. Service
- B. ReplicaSet
- C. pod
- D. Deployment
- E. StatefulSet

Answer: B

Explanation:

The *ReplicaSet* is responsible for ensuring that the desired number of Pods are running for a deployment. It monitors the Pods and automatically replaces failed Pods. Option 'A' is incorrect because services are for exposing applications, not for managing replicas. Option 'C' is incorrect because Pods are individual containers. Option 'D' is incorrect because Deployments use ReplicaSets to manage their replicas. Option 'E' is incorrect because StatefulSets are for managing Pods with unique identities and persistent storage, which is not required in this scenario.

Question: 10

You are using a Kubernetes deployment to run a web application. You want to roll out a new version of the application with minimal downtime. Which strategy can be used to achieve this?

- A. Recreate
- B. RollingUpdate
- C. BlueGreen
- D. Canary
- E. Immutable

Answer: B,C,D

Explanation:

RollingUpdate, BlueGreen, and Canary are all strategies that can be used to roll out new application versions with minimal downtime. RollingUpdate updates Pods one by one, ensuring that there are always healthy Pods running. BlueGreen creates two separate environments, one for the old version and one for the new version, and then switches traffic over to the new environment. Canary gradually rolls out the new version to a small subset of users before making it available to everyone. Option 'A' (Recreate) would completely stop the old Pods before creating new ones, which could lead to downtime. Option 'E' (Immutable) is not a deployment strategy for rolling out updates.



CERTSWARRIOR

FULL PRODUCT INCLUDES:

Money Back Guarantee



Instant Download after Purchase



90 Days Free Updates



PDF Format Digital Download



24/7 Live Chat Support



Latest Syllabus Updates



For More Information – Visit link below:

<https://www.certswarrior.com>

16 USD Discount Coupon Code: U89DY2AQ