Kubernetes interview questions

1. Kubernetes architecture?

Master node/control plane	Workload/workernode
Kube-Api Server	Kubelet
Scheduler	Container
Controller manager	pods
Etcd	Kube-proxy

2. What is deployment and stateful set?

A Kubernetes Deployment tells Kubernetes how to create or modify instances of the pods that hold a containerized application. Deployments can help to efficiently scale the number of replica pods, enable the rollout of updated code in a controlled manner, or roll back to an earlier deployment version if necessary.

You can use StatefulSets to deploy stateful applications and clustered applications that save data to persistent storage, such as Compute Engine persistent disks. StatefulSets are suitable for deploying Kafka, MySQL, Redis, ZooKeeper, and other applications needing unique, persistent identities and stable hostnames.

The key difference between stateful and stateless applications is that stateless applications don't “store” data. On the other hand, stateful applications require backing storage.

3. what is demonset ?

A Daemonset is another controller that manages pods like Deployments, ReplicaSets, and StatefulSets. It was created for one particular purpose: ensuring that the pods it manages to run on all the cluster nodes. As soon as a node joins the cluster, the DaemonSet ensures that it has the necessary pods running on it.

4. what is replicaset?

A ReplicaSet (RS) is a Kubernetes object that ensures there is always a stable set of running pods for a specific workload. The ReplicaSet configuration defines a number of identical pods required, and if a pod is evicted or fails, creates more pods to compensate for the loss.

5. What are secrets vs. config maps?

Secrets are confidential/sensitive data where as it is saved and encoding data (base64).

Config maps is un-confidential/non sensitive data.

6. What is storage class, persistent volume, Persistent volume claim?

A StorageClass provides a way for administrators to describe the "classes" of storage they offer.

A persistent volume is a piece of storage in a cluster that an administrator has provisioned.

A PersistentVolumeClaim (PVC) is a request for storage by a user.

7. What are taints and toleration?

Node affinity is a property of Pods that attracts them to a set of nodes (either as a preference or a hard requirement). Taints are the opposite -- they allow a node to repel a set of pods.

8. What is a tainted node?

9. What is recycle policy?

The Recycle reclaim policy is deprecated. Instead, the recommended approach is to use dynamic provisioning.

10. Pod lifecycle and different status of pod?

Pods follow a defined lifecycle, starting in the Pending phase, moving through Running if at least one of its primary containers starts OK, and then through either the Succeeded or Failed phases depending on whether any container in the Pod terminated in failure.

Pending, running, succeeded, failed, unknown

11. Lifecycle of a volume and claim?

Lifecycle stage of PV and PVC is

Provising, binding, using, reclaim, retain, deletion.

12. Access modes of pv?

ReadWriteOnce -- the volume can be mounted as read-write by a single node.

ReadOnlyMany -- the volume can be mounted read-only by many nodes.

ReadWriteMany -- the volume can be mounted as read-write by many nodes.

13. What is the control manager in Kubernetes?

The Kubernetes Controller Manager (also called kube-controller-manager) is a daemon that acts as a continuous control loop in a Kubernetes cluster. The controller monitors the current state of the cluster via calls made to the API Server, and changes the current state to match the desired state described in the cluster's declarative configuration.

The Kubernetes control plane consists of a core component called kube-controller-manager. This component is responsible for running multiple controllers that maintain the desired state of the cluster. These controllers are packaged together with the kube-controller-manager daemon.

14. What is Pod affinity and anti-affinity?

The Pod affinity rule tells the scheduler to place each replica on a node that has a Pod with the label app=store .

The Pod anti-affinity rule tells the scheduler to avoid placing multiple app=web-store servers on a single node.

15. What is node affinity and anti-affinity?

Node affinity is a set of rules used by the scheduler to determine where a pod can be placed. The rules are defined using custom labels on nodes and label selectors specified in pods. Node affinity allows a pod to specify an affinity (or anti-affinity) towards a group of nodes it can be placed on.

An anti-affinity rule tells the scheduler not to place the new pod on the same node if the label on the new pod matches the label on another pod. Anti-affinity allows you to keep pods away from each other.

The affinity feature consists of two types of affinity: Node affinity functions like the nodeSelector field but is more expressive and allows you to specify soft rules. Inter-pod affinity/anti-affinity allows you to constrain Pods against labels on other Pods.