1**.What is Kubernetes and why it is important?**

**2.What is difference between docker swarm and kubernetes?**

**3.How does Kubernetes handle network communication between containers?**

**4.How does Kubernetes handle scaling of applications?**

**5.What is a Kubernetes Deployment and how does it differ from a ReplicaSet?**

**6.Can you explain the concept of rolling updates in Kubernetes?**

**7.How does Kubernetes handle network security and access control?**

**8.Can you give an example of how Kubernetes can be used to deploy a highly available application?**

**9.What is namespace is kubernetes? Which namespace any pod takes if we don't specify any namespace?**

**10.How ingress helps in kubernetes?**

**11.Explain different types of services in kubernetes?**

**12.Can you explain the concept of self-healing in Kubernetes and give examples of how it works?**

**13.How does Kubernetes handle storage management for containers?**

**14.How does the NodePort service work?**

**15.What is a multinode cluster and single-node cluster in Kubernetes?**

**16.Difference between create and apply in kubernetes?**

1. Kubernetes automates operational tasks of container management and includes built-in commands for deploying applications, rolling out changes to your applications, scaling your applications up and down to fit changing needs, monitoring your applications, and more—making it easier to manage applications.
2. Docker Swarm offers automatic load balancing, while Kubernetes does not. However, it is easy to integrate load balancing through third-party tools in Kubernetes. Kubernetes: Services are made discoverable through a single DNS name. Kubernetes accesses container applications through an IP address or HTTP route.
3. Kubernetes defines a network model called the container network interface (CNI), but the actual implementation relies on network plugins. The network plugin is responsible for allocating internet protocol (IP) addresses to pods and enabling pods to communicate with each other within the Kubernetes cluster.
4. When deploying applications in Kubernetes, several design decisions must be considered. One of the most important aspects is how you can scale your application using Kubernetes and the technique to use. Scaling an application in Kubernetes requires a different approach than with other services. Kubernetes Autoscaling provides a mechanism to automatically scale up or down the number of pods of an application based on resource utilization or other user-defined triggers.
5. A Deployment in Kubernetes is a higher-level abstraction that represents a set of replicas of your application. It ensures that your desired number of replicas of your application are running and available.
6. Rolling Update implements automated, rolling updates for the Pods in the StatefulSet. RollingUpdate causes the controller to delete and recreate each of its Pod, and each Pod one at a time. It waits until an updated Pod is running and ready before to updating its predecessor.
7. Kubernetes ships an integrated Role-Based Access Control (RBAC) component that matches an incoming user or group to a set of permissions bundled into roles. These permissions combine verbs (get, create, delete) with resources (pods, services, nodes) and can be namespace-scoped or cluster-scoped.
8. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container. Automatic bin packing You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks.
9. In Kubernetes, namespaces provides a mechanism for isolating groups of resources within a single cluster. If no namespace is provided to a pod it accepts the namespace as “default”.
10. The Ingress concept lets you map traffic to different backends based on rules you define via the Kubernetes API. An API object that manages external access to the services in a cluster, typically HTTP. Ingress may provide load balancing, SSL termination and name-based virtual hosting.
11. ClusterIP. Exposes a service which is only accessible from within the cluster.

NodePort. Exposes a service via a static port on each node’s IP.

LoadBalancer. Exposes the service via the cloud provider’s load balancer.

ExternalName. Maps a service to a predefined externalName field by returning a value for the CNAME record.

1. The idea behind self-healing Kubernetes is simple: If a container fails, Kubernetes automatically redeploys the afflicted container to its desired state to restore operations. Kubernetes implements self-healing at the Application Layer.

This means that if your app is well containerized and a pod crashes, Kubernetes will work to reschedule it as soon as possible. Containers are made available for clients only if they are ready to serve.

The redeployment is subject to the availability of sufficient infrastructure.

Through its self-healing ability, Kubernetes is able to achieve the following:

Restart failed containers

Kill containers that do not respond to client requests

1. In Kubernetes, the most basic type of storage is non-persistent—also known as ephemeral. Each container has ephemeral storage by default—this storage uses a temporary directory on the machine that hosts the Kubernetes pod. It is portable, but not durable. Kubernetes supports multiple types of persistent storage.
2. The NodePort service serves as the external entry point for incoming requests for your app. The assigned NodePort is publicly exposed in the kubeproxy settings of each worker node in the cluster. Every worker node starts listening on the assigned NodePort for incoming requests for the service.
3. A single cluster configured with the resources detailed can meet scaling requirements to a certain degree. However, it will result in increased resource consumption; the cluster will have to manage the responsibility of handling all the scaling requirements for running workloads.

In contrast, having multiple Kubernetes clusters means that workloads can be distributed across different clusters that can scale according to their specific load. Furthermore, a multi-cluster model allows software teams to distribute their application across different regions which exponentially increases their availability.

1. The key difference between kubectl apply and create is that apply creates Kubernetes objects through a declarative syntax, while the create command is imperative. The command set kubectl apply is used at a terminal's command-line window to create or modify Kubernetes resources defined in a manifest file.