1. Orchestration tools, such as Kubernetes, play a key role in the server infrastructure for the modern applications.

(a) Explain how these tools help manage and scale application servers.

(b) Describe how orchestration tools facilitate automated deployment, scaling, and management of application servers.

A:

(a) They can automatically handle scheduling of containers (Pods) onto available nodes, Monitor container health and restart failed ones, Balance loads across nodes, Scale up/down application instances based on demand or resource utilization, Self-heal by rescheduling containers when nodes fail.

(b) Deployment — using Deployments and ReplicaSets to roll out new versions and roll back if needed.

Scaling — automatically adjusts the number of Pods using the Horizontal Pod Autoscaler.

Management — continuously reconciles the desired state (from YAML definitions) with the actual state in the cluster.

2. Explain the difference between a Pod, Deployment, and Service.

A:

Pod is the smallest deployable unit in Kubernetes. A Pod runs one or more tightly coupled containers (usually one). It is to encapsulates an application container and its resources.

Deployment is a higher-level abstraction that manages Pods using ReplicaSets, and it is to ensures the desired number of Pods are running; supports updates and rollbacks.

Service is a stable network endpoint that exposes a set of Pods. It can provides load balancing and service discovery for Pods.

3. What is a Namespace in Kubernetes? Please list one example.

A:

A Namespace is a virtual cluster inside a Kubernetes cluster used to organize and isolate resources (Pods, Services, Deployments, etc.).

It helps in multi-team or multi-project environments to avoid name conflicts and control access.

Example:

Default namespaces in Kubernetes:

default kube-system kube-public

4. Explain the role of the Kubelet. How do you check the nodes in a Kubernetes cluster? (kubectl command expected)

A:

Kubelet can:

An agent running on every node.

Ensures containers are running as specified in Pod definitions.

Reports node and Pod status to the control plane.

Manages communication between the node and Kubernetes API server.

**To check nodes in a Kubernetes cluster:**

kubectl get nodes

5. What is the difference between ClusterIP, NodePort, and LoadBalancer services?

A:

ClusterIP is default service type. Exposes the service only inside the cluster, and it can internal access only

NodePort is exposes the service on a static port (30000–32767) on each node’s IP. And it can be accessible from outside via <NodeIP>:<NodePort>

LoadBalancer is integrates with cloud provider’s load balancer (e.g., AWS ELB, GCP LB). And it can be external access through public IP.

6. How do you scale a Deployment to 5 replicas using kubectl?

A:

kubectl scale deployment <deployment-name> --replicas=5

7. How would you update the image of a Deployment without downtime?

A:

Use a **rolling update**:

kubectl set image deployment/<deployment-name> <container-name>=<new-image>

8. How do you expose a Deployment to external traffic?

A:

Use the kubectl expose command:

kubectl expose deployment <deployment-name> --type=LoadBalancer --port=80 --target-port=8080

9. How does Kubernetes scheduling decide which node a Pod runs on?

A:

The Kubernetes Scheduler chooses a node for each new Pod based on:

Resource availability (CPU, memory).

Node selectors / affinity rules.

Taints and tolerations.

Pod affinity/anti-affinity rules.

Custom scheduling policies.

It scores all available nodes and assigns the Pod to the most suitable one.

10. What is the role of Ingress and how does it differ from a Service?

A:

Ingress is to manage external HTTP/HTTPS access to Services. It can provides routing, TLS termination, virtual hosting. And it operates at Layer 7 (HTTP/HTTPS).

Service is to expose Pods within or outside the cluster. It can be simple network access (ClusterIP, NodePort, LoadBalancer). And it operates at Layer 4(TCP/UDP).