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.

Orchestration tools like Kubernetes manage and scale application servers by automating tasks like starting/stopping containers, distributing workloads across multiple servers , replacing failed instances, and adjusting the number of running instances based on demand or defined rules. This ensures applications are highly available and can handle varying loads efficiently.

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

They facilitate automated deployment by rolling out new application versions consistently. Scaling is automated by increasing/decreasing replica counts. Management includes self-healing (restarting failed containers), load balancing traffic, and managing storage/network resources, reducing manual intervention.

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

A Pod is the smallest deployable unit, containing one or more tightly coupled containers sharing resources. A Deployment manages the lifecycle of Pods, ensuring the desired number of replicas are running and healthy, enabling updates and rollbacks. A Service provides a stable network endpoint and load balancing for a set of Pods, allowing other applications to discover and communicate with them reliably.

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

A Namespace in Kubernetes is a virtual cluster within a physical cluster, used to partition resources and provide scope for names (like Pod/Service names). It allows multiple teams or projects to share a cluster without conflict. One example is the kube-systemnamespace, which houses system components like the Kubernetes control plane.

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

The Kubelet is an agent running on each node. Its primary role is to ensure containers described in PodSpecs (received from the API server) are running and healthy on its node.To check the nodes in a cluster, can use the kubectl to get nodes command.

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

ClusterIP exposes the Service on an internal IP within the cluster, accessible only from inside the cluster. NodePort exposes the Service on a static port on each Node's IP, making it accessible externally via <NodeIP>:<NodePort>. LoadBalancer creates an external load balancer that routes traffic to the Service, providing a stable external IP/DNS name.

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

Use the command: kubectl scale deployment/<deployment-name> --replicas=5.

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

Update the image without downtime by triggering a rolling update. Use the command kubectl set image deployment/<deployment-name> <container-name>=<new-image>. The Deployment controller gradually replaces old Pods with new ones running the updated image, maintaining service availability throughout the process.

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

Expose a Deployment to external traffic by creating a Service of type NodePort or LoadBalancer that targets the Pods managed by the Deployment. The kubectl expose deployment/<deployment-name> --type=<ServiceType> --port=<port>command is a common way to do this.

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

You expose a Deployment to external traffic by creating a Service of type NodePort or LoadBalancer that targets the Pods managed by the Deployment. The kubectl expose deployment/<deployment-name> --type=<ServiceType> --port=<port>command is a common way to do this.

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

Ingress manages external HTTP/HTTPS access to Services within the cluster, typically providing features like host/path-based routing, SSL termination, and load balancing. It acts as a layer 7 (application layer) entry point. A Service provides internal network connectivity and load balancing for Pods at layer 4 (TCP/UDP). Ingress usesServices (usually ClusterIP) to route traffic to the correct backend Pods; it doesn't replace them but provides a higher-level, more feature-rich way to expose HTTP(S) services externally.