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 such as Kubernetes help manage and scale application servers by automating the deployment and coordination of containers across multiple servers, ensuring that resources are efficiently used and that applications remain available even when demand changes. They monitor the health of services, restart failed containers, and balance workloads dynamically across nodes to maintain performance and reliability in distributed environments.

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

Orchestration tools also facilitate automated deployment, scaling, and management of application servers by using declarative configurations that define the desired system state, allowing the system to automatically adjust itself to match that state. They enable continuous integration and continuous deployment (CI/CD) pipelines, automatically scale resources based on real-time metrics, perform rolling updates without downtime, and provide centralized monitoring and logging for efficient maintenance and troubleshooting.

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

A **Pod** is the smallest deployable unit in Kubernetes that represents one or more containers running together on the same node and sharing storage, network, and configuration. A **Deployment** is a higher-level controller that manages Pods, ensuring the desired number of replicas are running, handling updates, rollbacks, and maintaining consistency across application instances. A **Service** provides a stable network endpoint to expose Pods to other services or users, automatically load balancing traffic across available Pods even when their underlying IP addresses change.

3. Whatis aNamespace in Kubernetes? Please list one example

A **Namespace** in Kubernetes is a logical partition within a cluster that allows you to organize and isolate resources such as Pods, Services, and Deployments. It helps manage multiple environments or teams within the same cluster by providing separation for resource names, access control, and resource quotas. For example, a company might create separate namespaces like **“development”**, **“testing”**, and **“production”** to isolate different stages of application deployment.

4. Explain the role of the Kubelet. How do you check the nodes in a Kubernetes cluster?

(kubectl command expected)

use the command: kubectl get nodes

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

A **ClusterIP** service exposes the application only inside the Kubernetes cluster, making it accessible to other Pods but not to external users. A **NodePort** service opens a specific port on each node's IP address, allowing external access to the application through <NodeIP>:<NodePort>. A **LoadBalancer** service automatically provisions an external load balancer (usually from a cloud provider) to distribute incoming traffic across multiple nodes and Pods, providing a single external IP address for access.

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

Scale a Deployment to 5 replicas using the following command:

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

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

Update the image of a Deployment without downtime by using Kubernetes' rolling update feature with the following command:

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

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** using the kubectl expose command. For example:

kubectl expose deployment <deployment-name>

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

Kubernetes scheduling decides which node a Pod runs on by evaluating available nodes and selecting one that meets the Pod's resource requirements and constraints. The scheduler considers factors such as CPU and memory requests, node taints and tolerations, labels and node selectors, affinity and anti-affinity rules, and overall node load. It then assigns the Pod to the most suitable node to balance workloads and optimize cluster resource utilization.

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

An **Ingress** in Kubernetes manages external access to services within the cluster, typically providing HTTP and HTTPS routing based on hostnames or paths. It allows you to define advanced routing rules, SSL termination, and load balancing using a single external IP or domain. A **Service**, on the other hand, exposes a set of Pods and provides stable internal or external networking, but it lacks the routing and traffic management capabilities of an Ingress. In short, a Service connects traffic to Pods, while an Ingress controls and routes external HTTP/S traffic to those Services.