In-Class Assessment2

1. Orchestration Tools

(a) How orchestration tools help manage and scale application servers:

- **Automated Scaling**: Automatically scale applications up/down based on CPU, memory, or custom metrics
- **Self-healing**: Automatically restart failed containers, replace/reschedule pods when nodes die
- **Load Distribution**: Distribute traffic evenly across multiple application instances
- **Resource Management**: Efficiently allocate CPU, memory, and storage resources
- **Rolling Updates**: Update applications without downtime through gradual rollout

(b) How orchestration tools facilitate automated deployment, scaling, and management:

- **Declarative Configuration**: Define desired state in YAML/JSON files, system reconciles actual state
- **Service Discovery**: Automatically discover and connect services within the cluster
- **Storage Orchestration**: Automatically mount storage systems of choice
- **Batch Execution**: Manage batch and CI workloads, replacing failed containers
- **Configuration Management**: Store and manage sensitive information and configuration

2. Difference between Pod, Deployment, and Service

- **Pod**: Smallest deployable unit, represents a single instance of a running process, can contain one or more containers
- **Deployment**: Manages the lifecycle of pods, provides declarative updates, rolling deployments, and rollback capabilities
- **Service**: Abstract way to expose an application running on a set of pods as a network service

3. Namespace in Kubernetes

A Namespace provides a mechanism for isolating groups of resources within a single cluster. Names of resources need to be unique within a namespace, but not across namespaces.

Example: kube-system - namespace for Kubernetes system components

4. Role of Kubelet and Checking Nodes

Kubelet Role: An agent that runs on each node in the cluster, ensuring containers are running in a Pod. It takes PodSpecs and ensures the described containers are running and healthy.

Command to check nodes

```
kubectl get nodes
```

5. Difference between Service Types

- **ClusterIP**: Exposes the service on a cluster-internal IP, only accessible within the cluster (default)
- **NodePort**: Exposes the service on each Node's IP at a static port, accessible from outside the cluster
- **LoadBalancer**: Exposes the service externally using a cloud provider's load balancer

6. Scaling a Deployment to 5 Replicas

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

7. Update Image Without Downtime

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

image>:<tag>

8. Expose Deployment to External Traffic

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

9. Kubernetes Scheduling

Resource requirements (CPU, memory)
Node affinity/anti-affinity rules
Taints and tolerations
Pod affinity/anti-affinity
Node selector labels
Resource availability

10. Role of Ingress vs Service

Service: Provides internal load balancing and service discovery within the cluster
Ingress: Manages external access to services, typically HTTP/HTTPS, providing:

- **SSL/TLS termination**
- **Name-based virtual hosting**
- **Path-based routing**
- **Load balancing**

Difference: Services provide L4 (TCP/UDP) load balancing, while Ingress provides L7 (HTTP/HTTPS) routing capabilities and acts as an API gateway for external traffic.