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

(a) How they help **manage and scale**

- Desired-state declaration: you write YAML that says "I want 7 copies of container X with 1 GB RAM each." The control loop constantly observes reality and makes adjustments (start, stop, reschedule) until the cluster matches the declaration.

- Horizontal auto-scaling: the Horizontal Pod Autoscaler watches CPU, memory, or custom metrics (QPS, queue length) every 15 s and increases/decreases replica count within user-defined min/max bounds.

- Cluster auto-scaling: when no node has enough free resources for pending pods, the cluster-autoscaler asks the cloud provider to launch a new VM; when nodes are under-utilised and pods can be moved, it drains and terminates VMs, saving cost.

- Self-healing: readiness probes remove unhealthy pods from load balancers; liveness probes restart containers; if a node disappears, all pods are re-created instantly on surviving nodes, keeping availability high without human intervention.

- Rolling updates & rollbacks: new container images are rolled out pod-by-pod with health checks; if a metric degrades, you can `--rollback` in seconds, eliminating maintenance windows.

(b) Automated **deployment, scaling, management**

- Deployment controller: keeps a declared number of identical pods, implements zero-downtime rolling updates (maxSurge, maxUnavailable), revision history and pause/resume.

- ReplicaSet: ensures the exact replica count; if a pod dies it is replaced immediately.

- DaemonSet: automatically runs log collectors, monitoring agents or network plugins on every node (or a subset).

- StatefulSet: gives stable hostnames, persistent volumes and ordered deployment for databases/zookeeper.

- Job/CronJob: run batch or scheduled work to completion.
- Configuration management: ConfigMaps and Secrets are distributed to pods automatically; when they change the Deployment can be reloaded without rebuilding images.
- Service discovery & load balancing: every pod gets a DNS name (`pod-ip.namespace.pod.cluster.local`) and services get a virtual ClusterIP (`service.namespace.svc.cluster.local`) with kube-proxy doing transparent load balancing.
- Declarative GitOps: combined with tools like ArgoCD or Flux, the same YAML stored in Git is continuously applied, giving full auditability and reproducible environments.

---

2. Difference between **Pod, Deployment, Service**

- **Pod**: smallest deployable Kubernetes object; 1-to-several containers that share network namespace (one IP) and storage volumes; ephemeral—when it dies it is not resurrected.
- **Deployment**: higher-level controller that manages ReplicaSets and therefore pods; offers rolling updates, rollback hooks, revision control, declarative scaling; ensures the desired number of healthy pods are always running.
- **Service**: stable virtual IP (and DNS) that never changes during the life of the service; uses label selectors to forward traffic to current set of healthy pods; can be internal (ClusterIP) or external (NodePort/LoadBalancer); survives pod churn.

---

3. **Namespace**

   A namespace is a logical boundary used to divide one physical cluster into multiple virtual clusters for different teams, environments, or projects. Resources inside a namespace are isolated by name; RBAC, network policies and resource quotas can be applied per namespace.
   Example:

```
kubectl create namespace staging
```

---

4. **Kubelet**
   The kubelet is the primary "node agent" that runs on every worker node. It:

- watches the API server for PodSpecs assigned to its node,
- invokes the container runtime (containerd, CRI-O, Docker) to start/stop containers,
- mounts volumes, downloads secrets,
- performs liveness/readiness probes and reports node status, pod status, and resource usage back to the control plane.
  Check nodes:
  ```
  kubectl get nodes -o wide
  ```

---

5. **ClusterIP vs NodePort vs LoadBalancer**

- **ClusterIP (default)**: virtual IP reachable only inside the cluster; used for service-to-service communication.
- **NodePort**: opens a static high port (30000–32767) on every node's IP and forwards traffic to the ClusterIP; quick way to expose something externally without a cloud LB.
- **LoadBalancer**: provisions an external load balancer (cloud provider or MetalLB) that has its own IP and forwards traffic to the service; simplest way to get production-grade ingress on public clouds.

---

6. Scale a Deployment to 5 replicas
   ```
   kubectl scale deployment/myapp --replicas=5
   ```

---

7. Update image without downtime
   ```
   kubectl set image deployment/myapp myapp-
   container=myregistry/myapp:v2.0
   ```
   Kubernetes performs a rolling update, respecting readiness probes; old pods are terminated only after new ones pass health checks, so availability is maintained.

---

8. Expose Deployment to external traffic
   `kubectl expose deployment myapp --type=LoadBalancer --port=80 --target-port=8080`
   Cloud clusters will provision an external LB IP; on-prem use `--type=NodePort` or create an Ingress resource.

---

9. How the **scheduler** decides
10. Filtering: remove nodes that don't satisfy resource requests (CPU, memory, GPU), node selectors, affinity/anti-affinity rules, taints & tolerations, data-locality for volumes.
11. Scoring: rank feasible nodes with priority functions (resource least requested, balanced resource allocation, spreading pods of the same service, node affinity weight, etc.).
12. Binding: the highest-scored node is written into the PodSpec; kubelet on that node starts the pod.

---

10. **Ingress vs Service**

- **Service** operates at L4 (TCP/UDP), gives a stable IP/port and round-robins packets to pods; one service usually fronts one application component.
- **Ingress** is an L7 HTTP/HTTPS API object implemented by an Ingress controller (NGINX, Traefik, HAProxy, GKE LB, AWS ALB). It provides host/path-based routing, TLS termination, redirects, rate-limiting, and can expose multiple services through a single external IP/hostname. Services are the backends; Ingress is the smart edge router in front of them.