Docker Mindmap:

1. **What is Docker and Why is it Used?**
   - A tool to package, ship, and run applications in lightweight containers.
   - Helps in consistent environment setup across different systems.
   - Eliminates "works on my machine" issues.
2. **Docker Images and Containers**
   - **Docker Image**: A blueprint for creating containers, like a template.
   - **Container**: A running instance of an image.
   - Containers are isolated but share the host OS.
3. **Dockerfile and How to Create It**
   - A script containing instructions to create a Docker image.
   - Common instructions:
     - FROM → Base image
     - RUN → Execute commands
     - COPY → Copy files into the image
     - CMD/ENTRYPOINT → Start the application
4. **Docker Networking Basics**
   - **Bridge Network**: Default, allows communication between containers.
   - **Host Network**: Uses host's network, no isolation.
   - **Overlay Network**: Used in Docker Swarm for multi-host communication.
   - **None**: No network, complete isolation.
5. **Docker Volumes**
   - Used to persist data beyond container lifecycle.
   - Types:
     - **Named Volumes**: Managed by Docker.
     - **Bind Mounts**: Direct mapping of host directory.
6. **Docker Compose**
   - A tool to define multi-container applications using a YAML file.

   - docker-compose.yml defines services, networks, and volumes.
   - Simplifies running multiple containers with a single command.

7. **Docker Registry**
   - Stores and distributes Docker images.
   - Public registry → Docker Hub.
   - Private registry → For internal use.

8. **Docker Swarm**
   - Native clustering tool for managing multiple Docker nodes.
   - Features:
     - Load balancing
     - Service scaling
     - High availability

9. **Docker Commands**
   - docker build → Create an image from a Dockerfile.
   - docker run → Start a container from an image.
   - docker ps → List running containers.
   - docker stop → Stop a running container.
   - docker rm → Remove a container.
   - docker rmi → Remove an image.
   - docker exec → Run commands inside a running container.
   - docker logs → View container logs.

10. **Docker vs Virtual Machines**

- **Docker**: Uses host OS kernel, lightweight, fast startup.
- **VMs**: Have a separate OS, heavy, slow startup.
- Docker is better for microservices and CI/CD, while VMs are useful for full OS emulation.

Kubernetes (K8s) Mind Map

1. **What is Kubernetes and Why is it Used?**
   - An open-source container orchestration platform.
   - Manages containerized applications automatically.
   - Provides scalability, high availability, and automation.
2. **Kubernetes Pods and Deployments**
   - **Pod**: The smallest unit in Kubernetes, contains one or more containers.
   - **Deployment**: Manages multiple replicas of pods and updates them without downtime.
3. **Services in Kubernetes**
   - Used to expose pods and enable communication.
   - Types:
     - **ClusterIP**: Internal access within the cluster.
     - **NodePort**: Exposes service on a static port of each node.
     - **LoadBalancer**: Uses an external cloud provider's load balancer.
4. **ConfigMaps and Secrets**
   - **ConfigMaps**: Store non-sensitive configuration data (e.g., environment variables).
   - **Secrets**: Store sensitive data like passwords and API keys in an encrypted format.
5. **Namespaces in Kubernetes**
   - Used to organize and isolate resources within a cluster.
   - Helps manage multi-team and multi-project environments.
6. **Kubernetes Volumes**
   - Provide persistent storage for pods.
   - Types:
     - **emptyDir**: Temporary storage, deleted when the pod stops.
     - **hostPath**: Uses a directory on the host node.

    - **PersistentVolume (PV) & PersistentVolumeClaim (PVC)**: For long-term storage across restarts.

7. **Autoscaling in Kubernetes**
    - Automatically adjusts resources based on demand.
    - Types:
        - **Horizontal Pod Autoscaler (HPA)**: Scales pods based on CPU/memory usage.
        - **Vertical Pod Autoscaler (VPA)**: Adjusts resource requests/limits of pods.
        - **Cluster Autoscaler**: Adjusts the number of nodes in a cluster.
8. **Role-Based Access Control (RBAC)**
    - Manages permissions for users and applications in the cluster.
    - Components:
        - **Roles & ClusterRoles**: Define access rules.
        - **RoleBindings & ClusterRoleBindings**: Assign roles to users or groups.
9. **Helm Charts (Package Management)**
    - Helm is a package manager for Kubernetes.
    - Uses **Charts**, which are pre-configured Kubernetes application templates.
    - Simplifies deployment and management of complex applications.
10. **Basic Kubernetes Commands**

- kubectl get pods → List running pods.
- kubectl get services → List services in the cluster.
- kubectl apply -f file.yaml → Deploy resources using a YAML file.
- kubectl delete pod pod-name → Delete a specific pod.
- kubectl describe pod pod-name → Get detailed information about a pod.
- kubectl logs pod-name → View logs of a container in a pod.
- kubectl scale deployment deployment-name --replicas=3 → Scale a deployment.