Docker and Kubernetes Overview

1. Docker: Brief Overview

Docker is a platform to package applications and dependencies into containers. Containers ensure the application runs the same in any environment.

2. Common Docker Commands

```
docker build -t <image_name>:<tag> .    # Build Docker image
docker images                           # List all images
docker ps -a                            # Show all containers
docker run -d -p 8080:80 <image>        # Run container
docker exec -it <container_id> bash     # Enter container
docker stop <container_id>              # Stop a container
docker rm <container_id>                # Remove container
docker rmi <image_id>                   # Remove image
docker login                            # Login to DockerHub
docker push <image>                     # Push image
docker pull <image>                     # Pull image
```

3. What is a Dockerfile?

A Dockerfile is a text file with instructions to build a Docker image. Each instruction creates a layer.

4. Project Structure

```
mern-app/
 backend/
    server.js
    package.json
    Dockerfile
 frontend/
    src/
    public/
    package.json
    Dockerfile
 docker-compose.yml
```

5. Dockerfile for Backend

```
FROM node:18-alpine
WORKDIR /usr/src/app
COPY package*.json ./
```

```
RUN npm install
COPY . .
EXPOSE 5000
CMD ["node", "server.js"]
```

6. Dockerfile for Frontend

```
FROM node:18-alpine
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
FROM nginx:alpine
COPY --from=0 /usr/src/app/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

7. Dockerfile Keywords (Instructions)

```
FROM       # Set base image
WORKDIR    # Set working directory
COPY       # Copy files to image
ADD        # Like COPY but supports URLs/archives
RUN        # Execute command
CMD        # Default command to run
ENTRYPOINT # Main command to run
EXPOSE     # Expose port
ENV        # Set environment variable
ARG        # Build-time variable
LABEL      # Add metadata
VOLUME     # Mount external volume
USER       # Set user
```

8. To build and run the Dockerfile

```
docker build -t mynodeapp .
docker run -p 3000:3000 mynodeapp
```

9. Kubernetes: Brief Overview

Kubernetes (K8s) is a container orchestration platform used to deploy, scale, and manage containers across

clusters.

10. Common Kubernetes Commands

```
kubectl create -f <file>.yaml                # Create resource
kubectl apply -f <file>.yaml                # Apply changes
kubectl get pods                            # List pods
kubectl get services                        # List services
kubectl describe pod <pod-name>             # Pod info
kubectl logs <pod-name>                     # Show logs
kubectl exec -it <pod-name> -- bash         # Open terminal
kubectl delete -f <file>.yaml               # Delete resource
kubectl scale deployment <name> --replicas=3
kubectl expose deployment <name> --type=LoadBalancer --port=80
```

11. What is a Kubernetes YAML File?

Kubernetes YAML files are used to define resources like Pods, Deployments, and Services in a declarative way.

12. Basic YAML Example (Deployment + Service)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
  labels:
    app: myapp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp-container
        image: myapp:latest
```

```
        ports:
        - containerPort: 5000
---
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
spec:
  type: NodePort
  selector:
    app: myapp
  ports:
  - port: 80
    targetPort: 5000
    nodePort: 30007
```

13. Explanation of YAML Keywords

```

apiVersion   # API version of resource type
kind         # Resource type (e.g. Deployment, Service)
metadata     # Metadata info (name, labels)
spec         # Desired state
replicas     # Number of pods
selector     # Match pods using labels
template     # Pod definition inside deployment
containers   # List of containers
image        # Docker image
ports        # Exposed ports
type         # Service type: NodePort, LoadBalancer
targetPort   # Port in container
nodePort     # Port exposed on node
```

14. File Usage

```

kubectl apply -f deployment.yaml   # Apply YAML
kubectl get deployments            # View deployments
kubectl get services               # View services
```