Week3-day4

📌 What is Kubernetes?

Kubernetes (also called **K8s**) is an **open-source container orchestration platform** that helps manage containerized applications efficiently across multiple nodes.

🚀 Why Do We Need Kubernetes?

Before Kubernetes, developers used **Docker** to create containers. But managing multiple containers manually became difficult. Kubernetes solves these problems by automating tasks like:

1. **Scaling** – Handles high traffic by increasing or decreasing containers automatically.
   (ek online shopping website hai, jisme **normal din** par 100 log aate hain, lekin **sale ke time** par 10,000 log ek saath aane lagte hain. Agar sirf **ek container (server)** ho, to wo **overload** ho jayega aur website slow ho sakti hai ya crash ho sakti hai.)

2. **Load Balancing** – Distributes traffic among multiple containers.
3. **Self-Healing** – Restarts crashed containers automatically.
4. **Zero Downtime Deployment** – Updates applications without stopping them.
5. **Service Discovery** – Assigns unique IPs to services so apps can find each other.

🛠 Kubernetes Core Concepts

🔹 Pod

- The **smallest deployable unit** in Kubernetes.
- A **Pod** contains one or more containers.
- Containers inside the same Pod **share storage and network**.

🔹 Node

- A **physical or virtual machine** where Kubernetes runs.
- Kubernetes clusters consist of **multiple nodes** (worker nodes).
- Each node runs a **kubelet** (an agent that manages containers on the node).

🔹 Cluster

- A group of **multiple nodes** running Kubernetes.
- The **Master Node** manages the cluster.
- **Worker Nodes** run application workloads.

☼ Kubernetes Architecture

1️⃣ Control Plane (Master Node)

The **brain of Kubernetes**, responsible for managing the cluster.

- **API Server** – The entry point for all commands.
- **Scheduler** – Assigns Pods to Nodes.
- **Controller Manager** – Monitors and manages different components.
- **etcd** – A key-value store that keeps track of cluster state.

2️⃣ Worker Nodes

Worker nodes run actual containerized applications. Each node has:

- **kubelet** – Ensures Pods are running on the node.
- **kube-proxy** – Manages network communication.
- **Container Runtime** – Runs containers (Docker, containerd, etc.).

⚡ Setting Up Kubernetes

```
#!/bin/bash
# Kubernetes Installation Script for WSL Ubuntu

set -e  # Exit immediately if a command fails

echo "===== Kubernetes Installation Script for WSL Ubuntu ====="

# Update and install dependencies
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common gnupg2
conntrack

# Install Docker
echo "[1/4] Installing Docker..."
sudo apt-get remove -y docker docker-engine docker.io containerd runc || true
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $
(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
sudo usermod -aG docker $USER
sudo docker run hello-world  # Test Docker

# Install kubectl
echo "[2/4] Installing kubectl..."
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
rm kubectl
kubectl version --client  # Test kubectl
```

Install Minikube
echo "[3/4] Installing Minikube..."
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
rm minikube-linux-amd64

Install Helm
echo "[4/4] Installing Helm..."
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

echo "===== Installation Complete! ====="
echo "To start Minikube, run: 'minikube start –driver=docker'"

Yeh sab packages system dependencies hain jo Kubernetes ko run karne ke liye chahiye hote hain.

🚀 Deploying a Flask App on Kubernetes

Step 1: Create the Flask App

Create a file `app.py`:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, Kubernetes!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Step 2: Create a Dockerfile

```
FROM python:3.9
WORKDIR /app
COPY app.py .
RUN pip install flask
CMD ["python", "app.py"]
EXPOSE 5000
```

Step 3: Build and Push the Docker Image

docker build -t my-flask-app .
docker tag my-flask-app my-dockerhub-user/my-flask-app:v1
docker push my-dockerhub-user/my-flask-app:v1

Step 4: Deploy to Kubernetes

Create a deployment `deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  name: flask-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app
    spec:
      containers:
      - name: flask-container
        image: my-dockerhub-user/my-flask-app:v1
        ports:
        - containerPort: 5000

Apply the deployment:
kubectl apply -f deployment.yaml
```

Output of a demo.sh file -