##################################
Kubernetes (K8s) Cluster Setup
##################################

A Kubernetes Cluster is a group of servers working together to run containerized applications. It can be set up in two main ways:

A Kubernetes Cluster = Control Plane + Worker Nodes + Pods + Resources + Networking + Storage

1) Self-Managed Cluster
In this setup, we manually install and manage Kubernetes on our own infrastructure.

a) MiniKube (Single Node)
-> Runs a single-node cluster on a local machine.
-> Best for learning and practicing Kubernetes concepts.
-> Not suitable for production as it lacks high availability and scalability.

b) Kubeadm (Multi-Node)
-> A tool for setting up a multi-node cluster manually.
-> Requires configuring the control plane, worker nodes, and networking.
-> Gives full control over the cluster but requires deep Kubernetes expertise.
-> Used for on-premise or customized Kubernetes deployments.

2) Cloud Provider-Managed Cluster (Pre-configured, ready-to-use)
Cloud providers offer fully managed Kubernetes services, where they handle cluster maintenance, updates, and availability.

a) AWS EKS (Elastic Kubernetes Service)
-> A managed Kubernetes service on Amazon Web Services.

b) Azure AKS (Azure Kubernetes Service)
-> Microsoft Azure's managed Kubernetes offering.

c) GCP GKE (Google Kubernetes Engine)
-> Google Cloud's fully managed Kubernetes solution.

#########################
MiniKube Setup
#########################

Step-1 : Setup Linux VM

Login into AWS Cloud account
Create Linux VM with Ubuntu AMI - t2.medium
Select Storage as 50 GB or more with 2 vCPU required minimum(Default is 8 GB only for Linux)
Create Linux VM and connect to it using SSH Client

Step-2 : Install Docker In Ubuntu VM

```
sudo apt update
curl -fsSL get.docker.com | /bin/bash
sudo usermod -aG docker ubuntu
exit
```

Step-3 : Updating system packages before installing Minikube dependencies

```
sudo apt update
sudo apt install -y curl wget apt-transport-https
```

Step-4 : Installing Minikube

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
minikube version
```

Step-5 : Install Kubectl (Kubernetes Client)

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
kubectl version -o yaml
```

Step-6 : Start MiniKube Server

```
minikube start — driver=docker
```

Step-7 : Check MiniKube status

```
minikube status
```

Step-8 : Access K8S Cluster

```
kubectl cluster-info
```

Step-9 : Access K8S Nodes

kubectl get nodes

$$ Setup Completed
$$

####################################
🚀 What is a POD in Kubernetes?
####################################

✅ Key Concepts Explained:
---
1. If you deploy an app, it will ultimately run inside one or more Pods. It is a building block to run app that we deploy in K8S

2. "Applications will be deployed as PODS in k8s."
Your app (e.g., a Spring Boot API) will be containerized using Docker. This container will then be wrapped inside a Pod and deployed on the cluster.

3. "To create PODS we will use Docker images."
A Pod runs one or more containers (usually one), and each container uses a Docker image. You can build a Docker image of your app and then deploy it inside a Pod.

4. "To create PODS we will use Manifest YML file."
A YAML manifest file defines the configuration for the Pod (or other resources like Deployments).

It includes:

a. The name of the Pod

b. The image to use

c. Ports to expose

c. Environment variables, etc.

5. "Create multiple PODS."
The same image (e.g., myapp:latest) can be used to create many Pods. This is how you scale your application—running multiple copies (Pods) to handle more traffic.

6. "If we run application with multiple pods then Load Balancing can be performed resulting in 99.9% uptime of the application."
High Availability: If one Pod crashes, others are still running, so your app stays available.Load Balancing: Kubernetes distributes traffic across Pods using a Service (like a load balancer).

7. "PODS count will be increased and decreased based on the demand (scalability)."
Kubernetes supports auto-scaling. You can scale Pods manually or automatically using a Horizontal Pod Autoscaler (HPA). Based on CPU/memory usage or custom metrics, Kubernetes will increase/decrease the number of Pods.

##################################
🚀 Kubernetes Services
##################################

-> A Kubernetes Service is used to expose a group of Pods so that they can be accessed reliably. Since Pods can be created and destroyed at any time (with changing IPs), a Service gives them a stable network identity.

🧭 Why Do We Use Services?

-> Pods are short-lived and can crash or restart.

-> Each time a Pod is created, it gets a new IP address.

-> Directly accessing Pods via IP is not reliable.

-> A Service gives a static IP to a group of Pods.

🌐 Types of Kubernetes Services

Kubernetes offers different types of services depending on how and where you want to expose your Pods:

🔹 1. ClusterIP (Default)
🔸 2. NodePort
🔹 3. LoadBalancer

🔐 ClusterIP Service (Internal Access Only)

📌 Key Points:
-> Pods are short-lived objects; if one crashes, Kubernetes replaces it with a new Pod.
-> Every new Pod gets a different IP address.

Note: 🛑 Never rely on Pod IPs to access an application.

-> A ClusterIP Service groups multiple Pods and assigns them a single static IP.

-> This static IP allows other components inside the cluster to access the group of Pods reliably, even when individual Pods change.

🚫 Access Scope:

-> Only accessible within the Kubernetes cluster.

-> Not reachable from the outside world (internet or external clients).

💡 Use Case:

-> Internal services such as databases, backend APIs, authentication services, etc.

Example: You don’t want to expose a database Pod to the internet, so you use a ClusterIP service to allow access only from other internal Pods.

🌐 What is a NodePort Service in Kubernetes?

-> A NodePort service is a type of Kubernetes Service that exposes your Pods outside the cluster using a port on each worker node (called a "NodePort").

🧭 Why Use NodePort?

By default, Pods and ClusterIP services are only accessible within the cluster.

NodePort makes them accessible externally by opening a static port (from 30000 to 32767) on each worker node.

It allows you to access your application using:

http://<NodeIP>:<NodePort>

Note: Here all traffic is routed to one worker node. Means load balancing cannot happen here.

🌐 What is a LoadBalancer Service in Kubernetes?

-> It not only provides external access to your app but also handles automatic traffic distribution across the backend Pods running on different worker nodes.