KUBERNETES

Kubernetes (K8s) is an open-source platform to manage containerized applications across clusters of nodes. It automates deployment, scaling, and operations.

Features:

- **Container Orchestration:** Kubernetes manages the lifecycle of containers, including their deployment, scaling, and updates.
- **Scalability and Resilience:** Kubernetes ensures that applications can scale up or down based on demand and can automatically recover from failures, ensuring high availability.
- **Service Discovery:** Kubernetes provides mechanisms for services to discover each other, making it easy to connect different parts of a distributed application.
- **Configuration Management:** Kubernetes offers tools like Secrets and ConfigMaps to securely manage application configurations and sensitive data.
- **Automation:** Kubernetes automates many of the manual tasks involved in managing containerized applications, reducing operational overhead.

📌 Core Concepts

1. Cluster

- A **Kubernetes cluster** is a set of nodes grouped together to run containerized applications.
- It consists of a **control plane (master node)** and one or more **worker nodes**.

2. Node

- A **node** is a virtual or physical machine in the cluster.
- Each node runs pods and is managed by the control plane.
- Two types:

- **Master Node**
- **Worker Node**

3. Master Node (Control Plane)

- Manages the cluster.
- Responsible for:
  - Scheduling pods
  - Maintaining cluster state
  - Monitoring and responding to events
- **Key components:**
  - kube-apiserver: Frontend of the control plane
  - etcd: Key-value store for cluster data
  - kube-scheduler: Assigns pods to nodes
  - kube-controller-manager: Runs controllers (like node controller, replication controller)

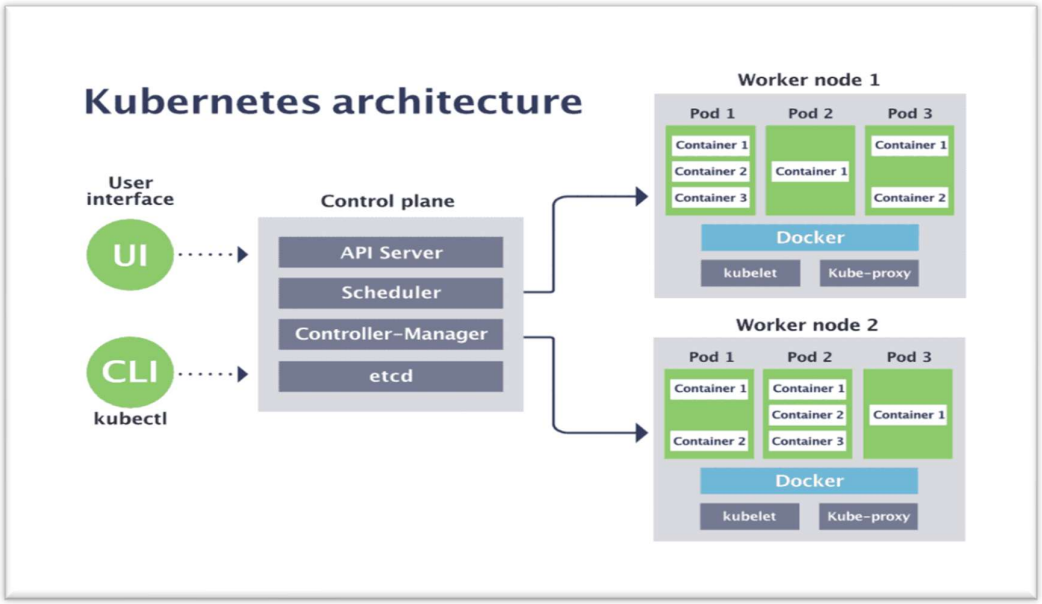
4. Worker Node

- Executes the containers.
- Managed by the master node.
- **Components:**
  - kubelet: Communicates with the control plane
  - kube-proxy: Network communication
  - Container Runtime (e.g., containerd or Docker)

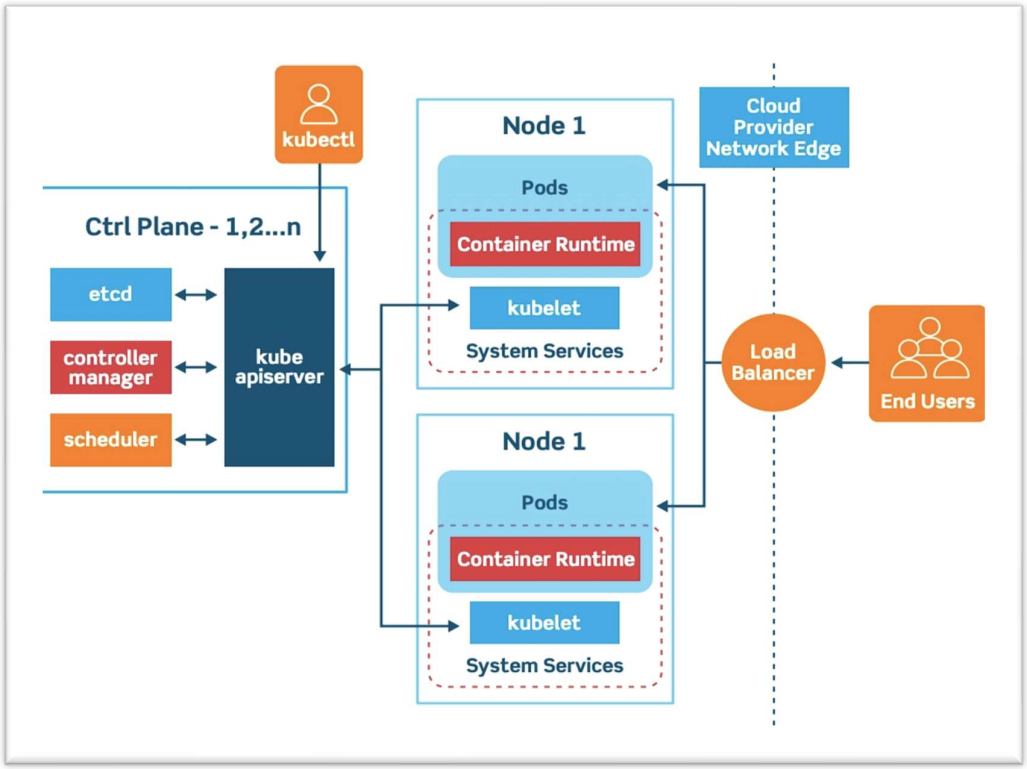
5. Pods

- The smallest deployable unit in Kubernetes.
- A pod contains one or more containers with shared storage/network.
- Pods are ephemeral (short-lived).

Kubernetes Architecture



Working:



☁️ Azure Kubernetes Service (AKS)

- **AKS** is a managed Kubernetes service in **Microsoft Azure**.
- Azure handles:
  - Control plane (master nodes)
  - Upgrades
  - Monitoring
- You manage:
  - Worker nodes
  - Deployments
  - Networking

Benefits of AKS:

- Simplified deployment
- Integrated with Azure Monitor and AAD
- Built-in scaling & self-healing

⚙️ kubectl – Kubernetes CLI Tool

Used to interact with Kubernetes clusters.

🔧 Basic Commands

Command	Description
kubectl get nodes	List all nodes in the cluster
kubectl get pods	List all running pods
Kubectl get pod --namespace=<name>	List all running pods in the given namespace
Kubectl create namespace <name>	Creates a namespace with given name

Command	Description
Kubectl run <pod-name> --image=<img-name>	Creates a pod for an image in docker hub's public repository
kubectl get services	List all services
Kubectl get namespaces	List all namespaces in the cluster
kubectl apply -f <file.yaml>	Apply configuration from a YAML file
kubectl create -f <file.yaml>	Create resources from YAML
kubectl delete -f <file.yaml>	Delete resources from YAML
kubectl describe pod <pod-name>	Detailed info about a pod
kubectl logs <pod-name>	View logs from a pod
kubectl exec -it <pod-name> -- /bin/bash	Access a running container inside a pod

🔗 AKS Specific Commands

1. **Login to Azure CLI**

   az login

2. **Get Credentials for Your AKS Cluster**

   az aks get-credentials --resource-group <resource-group-name> --name <aks-cluster-name>

3. **Check Cluster Context**

   kubectl config get-contexts

   kubectl config use-context <context-name>

📄 Kubernetes Manifest Files

Kubernetes uses **YAML manifest files** to define the desired state of cluster resources. These files declare what you want (e.g., a pod, a service, a deployment), and Kubernetes takes care of making it happen.

Each YAML file typically includes:

- apiVersion
- kind (resource type - object like pod)
- metadata (name, labels)
- spec (specifications- containers,ports)

🧱 1. Pod Manifest File

A **Pod** is the smallest deployable unit that runs one or more containers.

✅ **Sample pod.yaml**

```
apiVersion: v1
kind: Pod
metadata:
  name: my-nginx-pod
  labels:
    app: nginx
spec:
  containers:
    - name: nginx-container
      image: nginx
      ports:
        - containerPort: 80
```

🚀 **Command to Apply:**

```
kubectl apply -f pod.yaml
```

🌐 2. Service Manifest File

A **Service** exposes your pod(s) to other services or external users. Common types:

- ClusterIP (internal)
- NodePort (external on node port)
- LoadBalancer (for cloud-based external access)

✅ **Sample service.yaml**

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

🚀 **Command to Apply:**

```
kubectl apply -f service.yaml
```

🔁 3. Deployment Manifest File

A **Deployment** manages pods by maintaining the desired number of replicas, rolling updates, etc.

✅ **Sample deployment.yaml**

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

```
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
```

🚀 **Command to Apply:**

```
kubectl apply -f deployment.yaml
```

⚙️ Working Process: Master Node to Worker Node in Kubernetes

1. User Applies Manifest

```
kubectl apply -f deployment.yaml
```

- This command goes to the **API Server** on the **Master Node**.

2. API Server

- Validates the request.
- Updates the **etcd** database with the desired cluster state.

3. Scheduler

- Checks which **Worker Node** has available resources (CPU, memory).

- Chooses the best node and schedules the **pod** there.

4. Controller Manager

- Ensures the desired state is achieved.
- It sees: "We want 3 pods of NGINX" → ensures they are always running.
- If any pod fails, it reschedules them.

5. Kubelet (on Worker Node)

- Master node instructs **Kubelet** (agent on the Worker).
- Kubelet receives pod specs and creates the container via the **container runtime** (Docker, containerd).

6. Container Runtime

- Pulls the image (e.g., nginx) from a container registry (like Docker Hub).
- Runs the container inside a pod.

7. kube-proxy

- Manages networking rules to route requests to the correct pod.
- Handles load balancing for services.

🔁 **Ongoing Communication**

- **Health Checks:** Kubelet sends regular status reports to the Master.
- **Events:** If a pod dies, Master is notified → Controller reschedules it.
- **Scaling:** User changes replicas in deployment.yaml → Scheduler creates or deletes pods accordingly.

Some kubectl commands:

```
gouth@Priya_gowtami MINGW64 ~
$ kubectl get namespace
NAME                STATUS   AGE
default             Active   2d17h
gani-kube           Active   22h
jyoshna-meghadri    Active   17h
kube-node-lease     Active   2d17h
kube-public         Active   2d17h
kube-system         Active   2d17h
madhusrivamisett    Active   15h
poc-o1              Active   4h25m
priya-ns            Active   15h
reshma-node         Active   17h
testns              Active   38h

gouth@Priya_gowtami MINGW64 ~
$ kubectl create namespace example-ns
namespace/example-ns created

gouth@Priya_gowtami MINGW64 ~
$ kubectl get namespace
NAME                STATUS   AGE
default             Active   2d17h
example-ns          Active   5s
gani-kube           Active   22h
jyoshna-meghadri    Active   17h
kube-node-lease     Active   2d17h
kube-public         Active   2d17h
kube-system         Active   2d17h
madhusrivamisett    Active   15h
poc-o1              Active   4h27m
priya-ns            Active   15h
reshma-node         Active   17h
testns              Active   38h
```

```
gouth@Priya_gowtami MINGW64 ~
$ kubectl run nginx --image=nginx --namespace=example-ns
pod/nginx created

gouth@Priya_gowtami MINGW64 ~
$ kubectl get pod --namespace=example-ns
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          16s

gouth@Priya_gowtami MINGW64 ~
$ kubectl get service
NAME          TYPE           CLUSTER-IP     EXTERNAL-IP
kubernetes    ClusterIP      10.0.0.1       <none>
nginx-svc     LoadBalancer   10.0.14.236    135.235.248.95

gouth@Priya_gowtami MINGW64 ~
$ kubectl delete pod nginx --namespace=example-ns
pod "nginx" deleted
gouth@Priya_gowtami MINGW64 ~
$ kubectl get pod --namespace=example-ns
No resources found in example-ns namespace.

gouth@Priya_gowtami MINGW64 ~
$ kubectl delete namespace example-ns
namespace "example-ns" deleted

gouth@Priya_gowtami MINGW64 ~
$ kubectl get namespace
NAME                STATUS   AGE
default             Active   2d17h
gani-kube           Active   22h
jyoshna-meghadri    Active   17h
kube-node-lease     Active   2d17h
kube-public         Active   2d17h
kube-system         Active   2d17h
madhusrivamisett    Active   15h
poc-o1              Active   4h34m
priya-ns            Active   16h
reshma-node         Active   17h
testns              Active   38h
```