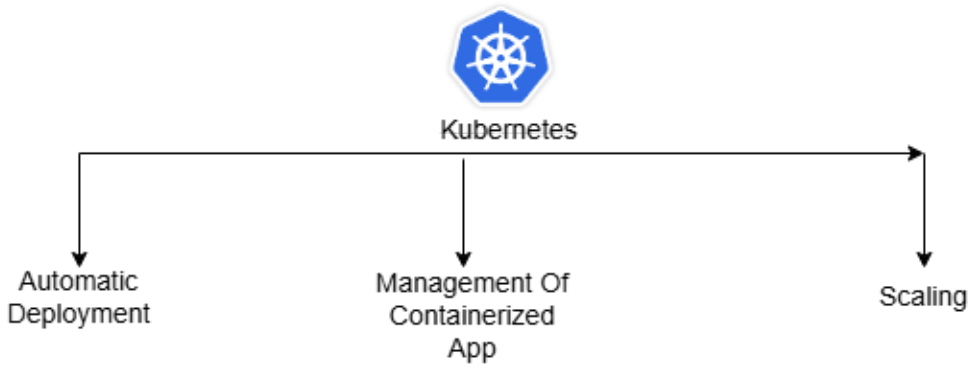
KUBERNETES BASICS AND INSTALLATION

What is kubernetes ?

Kubernetes (often abbreviated as K8s) is an open-source container orchestration platform. It automates the deployment, scaling, and management of containerized applications.

It was originally developed by Google and is now maintained by the Cloud Native Computing Foundation (CNCF).



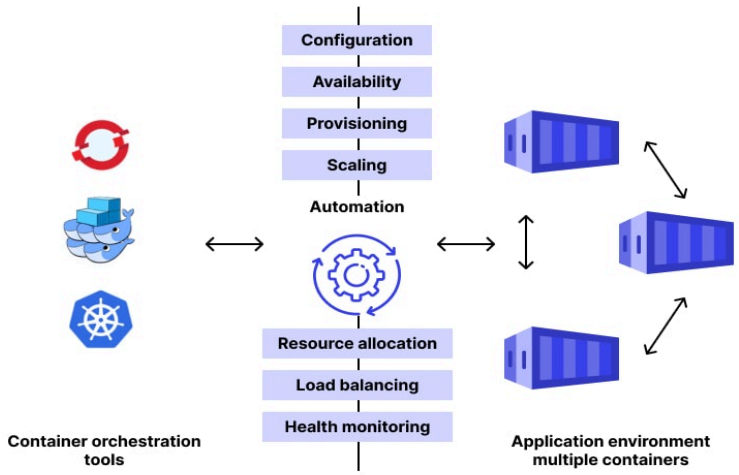
What is Container Orchestration?

Container orchestration is the process of automating the deployment, management, scaling, and networking of containers.

When you're running dozens or hundreds of containers (e.g., Docker containers) across many servers, manually managing them becomes impossible. Orchestration tools like Kubernetes, Docker Swarm, and Apache Mesos help solve this problem.

Key Functions of Container Orchestration

1. Deploy containers automatically
2. Scale containers up/down based on demand
3. Restart failed containers
4. Load balancing between containers
5. Service discovery (make containers find each other)
6. Manage secrets and configurations



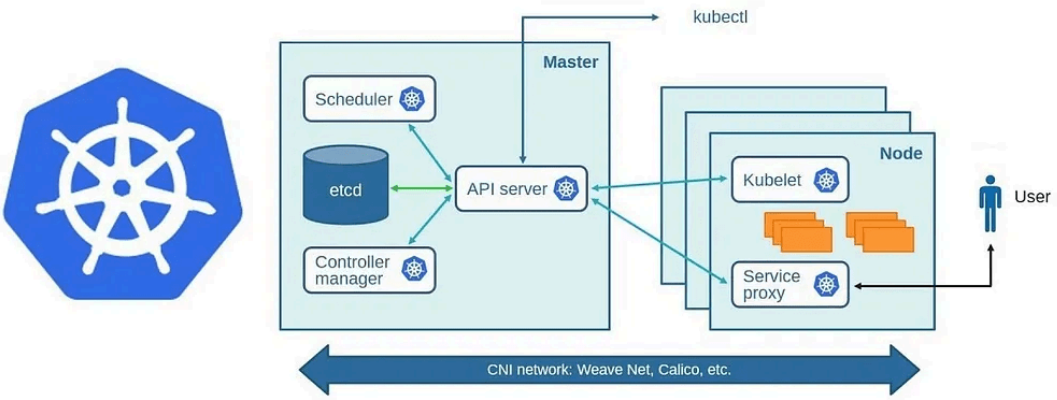
Kubernetes Use Cases

1. Auto-Scaling Apps – Scale up/down based on traffic or load.
2. Zero Downtime Deployments – Smooth rolling updates & rollbacks.
3. Microservices Management – Run and manage modular services easily.
4. Multi-Cloud & Hybrid Cloud – Deploy across multiple clouds or on-prem.
5. CI/CD Pipelines – Automate testing, staging, and deployment.
6. Secure Secrets & Configs – Manage passwords & configs safely.
7. ML/AI Workloads – Train models & process data at scale.
8. Cost Optimization – Use only the resources you need.
9. Edge & IoT Computing – Run apps on edge devices with central control.
10. Game Server Hosting – Spin up/down game servers on demand.

Kubernetes Architecture Overview

Kubernetes follows a **Master-Worker** architecture:

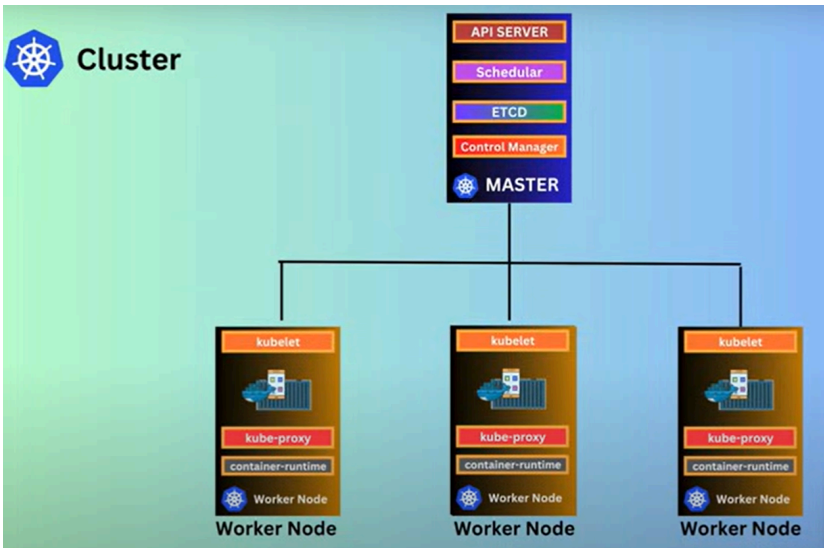
Kubernetes



Cluster :

A Kubernetes cluster is a group of machines (called nodes) that work together to run containerized applications automatically.

It’s the core setup where your apps live and run inside containers.



Worker Node Components

kubelet

The *"manager"* for each worker node. It ensures all containers on the node are healthy and running as expected.

kube-proxy

Acts like a *"traffic cop"* for network communication—both between Pods and from external clients to Pods. It routes network traffic appropriately.

Container Runtime

This is the software used to run containers. Common options include **Docker** and **containerd**.

Other Components

Pod

The smallest unit in Kubernetes. A Pod is a group of one or more containers that share resources. Think of it like an apartment in an apartment building.

Service

Like a *"phone directory"* for Pods. Since Pods can come and go, a Service provides a stable address for other parts of your app to find and communicate with them.

Volume

Acts like an *external hard drive* attached to a Pod to store data, ensuring it persists even if the container restarts.

Namespace

Used to divide cluster resources among multiple users or teams. Think of it like having different folders on a shared computer—each team gets their own space.

Ingress

The *"front door"* for external access to your apps. It controls how HTTP and HTTPS traffic is routed to your services.

Components of Kubernetes Master Node (Control Plane)

1. API Server

- **Role:** The entry point to the Kubernetes control plane.
- **Function:**

  Handles REST API requests (kubectl, dashboards, etc.)
  Validates and processes cluster operations

  Think of it as: The *front desk* or *gateway* to the cluster.

2. etcd

- **Role:** A distributed key-value **store**.
  Function:
  Stores all cluster state and configuration data
  Highly available and consistent

  Think of it as: The *database* of Kubernetes.

3. Scheduler

- **Role:** Assigns Pods to Nodes.
  Function:
  Watches for new Pods that need placement
  Picks the best Node based on resources, affinity rules, etc.

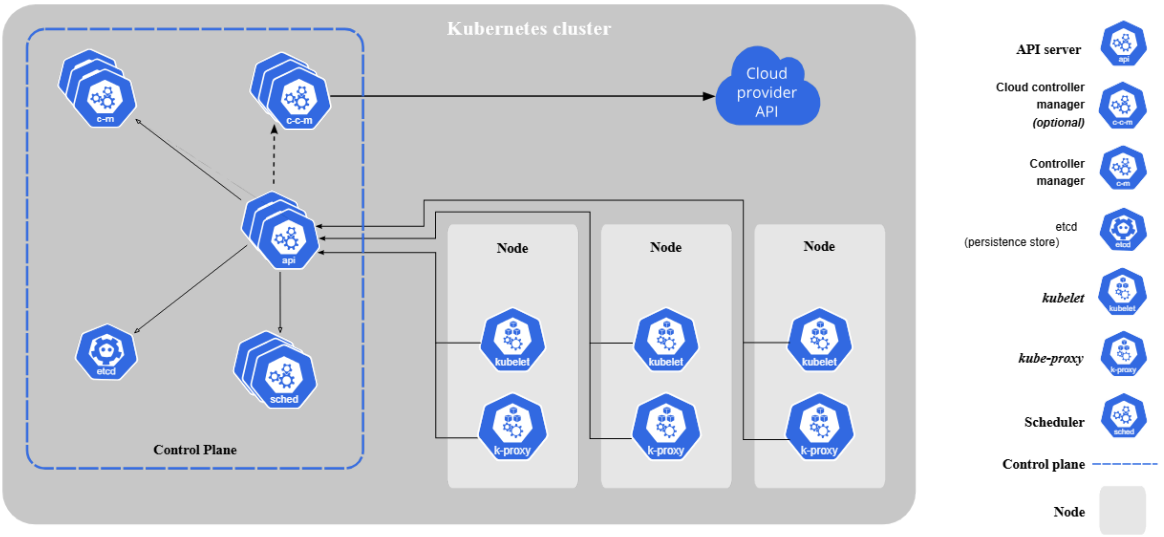
  Think of it as: The *event planner*—decides where things go.

4. Controller Manager

- **Role:** Runs background **controllers** to maintain cluster state.
  Function:
  Node Controller: Monitors node health
  Replication Controller: Ensures desired Pod counts
  Job Controller, Endpoint Controller, etc.

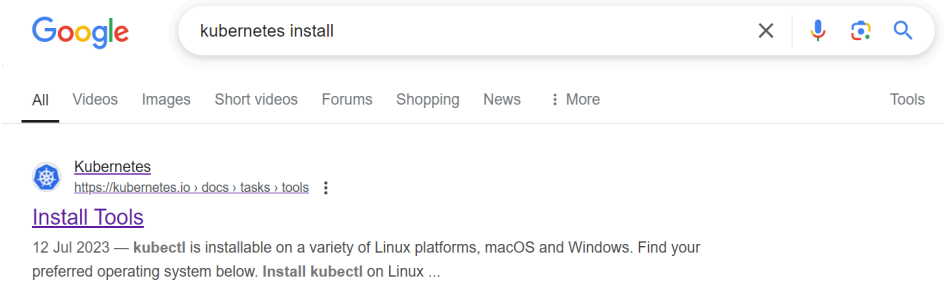
  Think of it as: A set of *automated bots* fixing and managing the system.

Components Of kubernetes

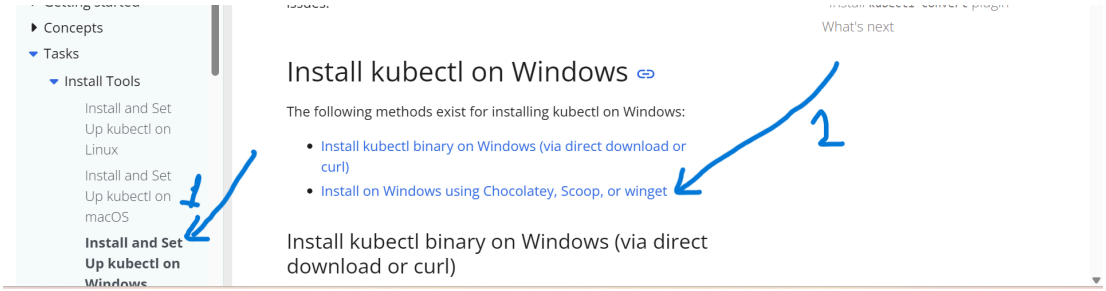


Kubernetes Installation :

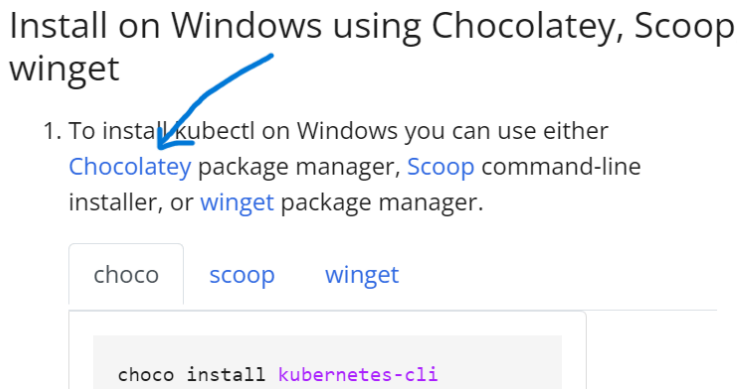
- Type Install Kubernetes and choose the Install Tools



- Then choose windows option like “Install and Set Up kubectl on Windows” left side of window
- Install kubectl for that choose option “Install on Windows using Chocolatey, Scoop, or winget”.



- Click on chocolatey



1. Install Chocolatey (Skip if already installed)

Open **PowerShell as Administrator** and run:

Set-ExecutionPolicy Bypass -Scope Process -Force; `

[System.Net.ServicePointManager]::SecurityProtocol = `

[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; `

iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))

---

2. Install kubectl (Kubernetes CLI)

choco install kubernetes-cli -y

Verify installation:

kubectl version --client

---

3. Install Minikube

choco install minikube -y

---

4. Install or Enable a Hypervisor

Choose one of the following:

- **Docker Desktop**: https://www.docker.com/products/docker-desktop

- **Hyper-V** (for Windows Pro/Edu):
  Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All
- **VirtualBox**: https://www.virtualbox.org
- **WSL2**: Let me know if you want setup steps for this.

5.Start Minikube

minikube start --driver=docker (if its not worked start using Hyper-V)

Using **Hyper-V**
minikube start --driver=hyperv

---

6. Verify Your Kubernetes Cluster

kubectl get nodes

You can check the mi*nikube status*

Minikube stat

Basics Commands Used

1. *minikube start/delete* —-> Build a single node cluster

   minikube status —-> show the status of your cluster

   minikube dashboard —> creates a web UI dashboard where we can monitor the states of cluster,pods deployment

   kubectl create deployment my-app --image-link → Creates a deployment using dockerhub image

   kubectl get deployments —--> *Shows the deployments*

   kubectl get pods —-> *It shows us pods available*

   kubectl delete deployment my-app —-> *It deletes the deployments*