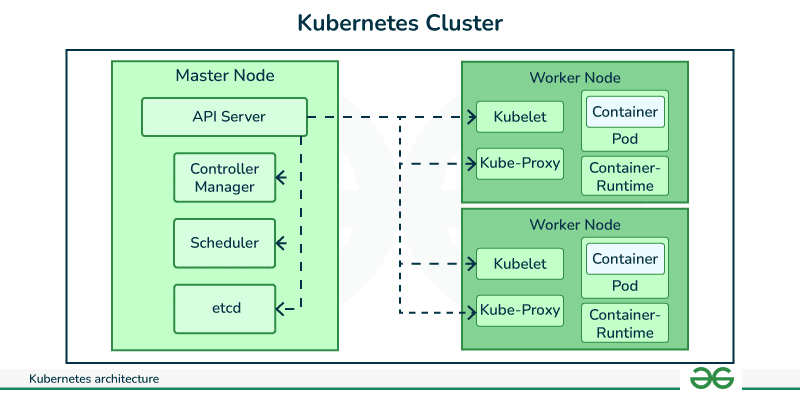
## **Kubernetes (often shortened as K8s) is an open-source container orchestration platform. It was originally developed by Google and is now maintained by the Cloud Native Computing Foundation (CNCF). But what does that mean in practice? Let’s break it down 🔹 Why Kubernetes? Modern applications often run inside containers (like Docker). Containers are lightweight and portable, but managing them at scale is hard: • How do you start hundreds of containers? • How do you scale them when traffic spikes? • What if some containers crash?**

**Kubernetes solves all of this. 💡**🔹 **What Kubernetes Does:** • ✅ Automates Deployment – start and manage containers easily.  
 • ✅ Scales Applications – add/remove replicas based on demand.  
 • ✅ Self-Heals – restarts failed containers automatically.  
 • ✅ Manages Networking & Load Balancing – routes traffic to healthy Pods.  
 • ✅ Portability – works across on-premises or cloud environments.

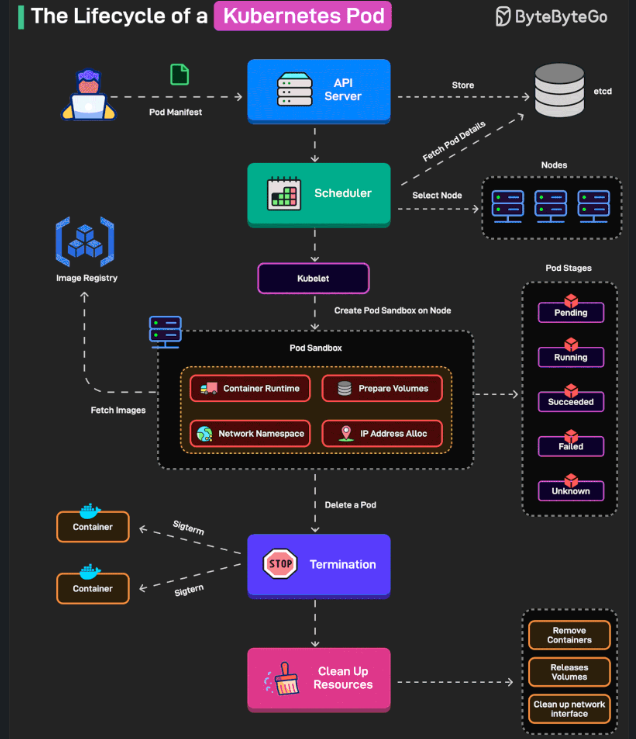
## **Kubernetes Cluster**

A group of nodes (Computers/Operating systems) working together to communicate with the help of Kubernetes software is known as the Kubernetes cluster. It works in a mechanism in such a way that it performs actions on the worker node with the help of the manager node.



1. **KubeProxy**: It is a resource in kubernetes cluster that helps in setting up the network setup throughout the cluster making the nodes able to communication with each other.
2. **Kube API server:**It is a resource in Kubernetes it takes the api requests from the other nodes and passes to Kubernetes controller to perform the specified actions. It works in the front line of the Kubernetes manager node.
3. **Kubernetes Controller**: It monitor the metrics of the nodes in the Kubernetes cluster ie., cpu , ram , storage other hard resources of the nodes in the cluster. Based on the monitoring when there is a need of launching the new pods , It specifies Kubernetes scheduler where to launch the pods , based on resources quota as required on checking the availability as per need on the nodes.
4. **Kube Scheduler:**It is a kind of resource in kubernetes Manager , that help in scheduling the pods by making request to the backend container engine ie., containerd of the particular node.

## **Kubernetes POD Life Cycle**



**The Lifecycle of a Kubernetes Pod**Every Pod in Kubernetes goes through a journey:  
 1️⃣ Pod Creation → API Server receives the manifest & stores it in etcd  
 2️⃣ Scheduling → Scheduler picks the best node  
 3️⃣ Pod Sandbox → Kubelet sets up runtime, volumes, network, and IP  
 4️⃣ Running Stages → Pending → Running → Succeeded/Failed/Unknown  
 5️⃣ Termination → SIGTERM sent, containers stop gracefully  
 6️⃣ Clean Up → Volumes released, network cleaned, containers removed

**Co𝐦**𝐦𝐨𝐧 𝐊𝐮𝐛𝐞𝐫𝐧𝐞𝐭𝐞𝐬 𝐟𝐢𝐥𝐞 𝐭𝐲𝐩𝐞𝐬 𝐲𝐨𝐮’𝐥𝐥 𝐮𝐬𝐞 👇  
  
📄 𝐩𝐨𝐝.𝐲𝐚𝐦𝐥 → Defines one or more containers with shared network and storage.  
🚀 𝐝𝐞𝐩𝐥𝐨𝐲𝐦𝐞𝐧𝐭.𝐲𝐚𝐦𝐥 → Manages scaling and rolling updates for Pods automatically.  
🔗 𝐬𝐞𝐫𝐯𝐢𝐜𝐞.𝐲𝐚𝐦𝐥 → Connects apps with a stable network name inside the cluster.  
⚙️ 𝐜𝐨𝐧𝐟𝐢𝐠𝐦𝐚𝐩.𝐲𝐚𝐦𝐥 → Stores configuration settings separately from your code.  
🔑 𝐬𝐞𝐜𝐫𝐞𝐭.𝐲𝐚𝐦𝐥 → Safely stores sensitive data like passwords or API keys.  
🌐 𝐢𝐧𝐠𝐫𝐞𝐬𝐬.𝐲𝐚𝐦𝐥 → Controls external access to apps (e.g., HTTP/HTTPS routing).  
💾 𝐩𝐯𝐜.𝐲𝐚𝐦𝐥 → Used to get storage that keeps data even after a Pod restarts.  
📈 𝐡𝐩𝐚.𝐲𝐚𝐦𝐥 → Auto-scales Pods up or down based on resource usage.  
👤 𝐬𝐞𝐫𝐯𝐢𝐜𝐞𝐚𝐜𝐜𝐨𝐮𝐧𝐭.𝐲𝐚𝐦𝐥 → Grants Pods permission to access Kubernetes APIs.  
📝 𝐫𝐨𝐥𝐞.𝐲𝐚𝐦𝐥 → Defines access rules for specific actions within a namespace.  
🔗 𝐫𝐨𝐥𝐞𝐛𝐢𝐧𝐝𝐢𝐧𝐠.𝐲𝐚𝐦𝐥 → Assigns a Role to a user or service account in a namespace.  
🌍 𝐜𝐥𝐮𝐬𝐭𝐞𝐫𝐫𝐨𝐥𝐞.𝐲𝐚𝐦𝐥 → Defines access rules that apply across the entire cluster.  
🌀 𝐝𝐚𝐞𝐦𝐨𝐧𝐬𝐞𝐭.𝐲𝐚𝐦𝐥 → Ensures a Pod runs on every node (e.g., logging/monitoring agents).  
🎯 𝐣𝐨𝐛.𝐲𝐚𝐦𝐥 → Runs one-time tasks until they complete successfully.  
🚫 𝐧𝐞𝐭𝐰𝐨𝐫𝐤𝐩𝐨𝐥𝐢𝐜𝐲.𝐲𝐚𝐦𝐥 → Restricts and controls which Pods can communicate.

**To start with, read about monolithic and microservice architecture**

|  |  |
| --- | --- |
| Monolithic Vs Microservice Architecture for applications | 1. Monolithic vs Microservice architecture  <https://www.youtube.com/watch?v=lL_j7ilk7rc&t=189s>  2. What, Why use Microservices.  <https://www.youtube.com/watch?v=rv4LlmLmVWk&t=1041s> |

**What is Kubectl?**

kubectl is the command-line tool for interacting with Kubernetes clusters. It allows users to deploy applications, inspect and manage cluster resources, and view logs.

**Practical:**

1. Enable Kubernetes Service in Docker Desktop.
2. This pulls some images of Kubernetes cluster and kubectl in Docker Desktop.
3. Open CMD and run these commands to check kubernetes resources.
4. **kubectl version**
5. **kubectl get nodes**
6. **kubectl get nodes -o wide**
7. **kubectl get pods**
8. **kubectl get deployments**
9. **kubectl get services**

**Running First POD**

|  |  |
| --- | --- |
| Running Pod on the cluster | # To run pod with nginx container  kubectl run webapp --image=nginx:latest  # To check the resources created  # kubectl get <resource-name>  kubectl describe pods webapp  # Use port-forward command -  kubectl port-forward webapp 8082:80  **Please note** - kubectl port-forward allows using resource name, such as a pod name, to select a matching pod to port forward |

Every resource by default is created in the **default namespace**.

To create a new namespace:

* **kubectl create namespace monitoring**
* **kubectl get namespaces**