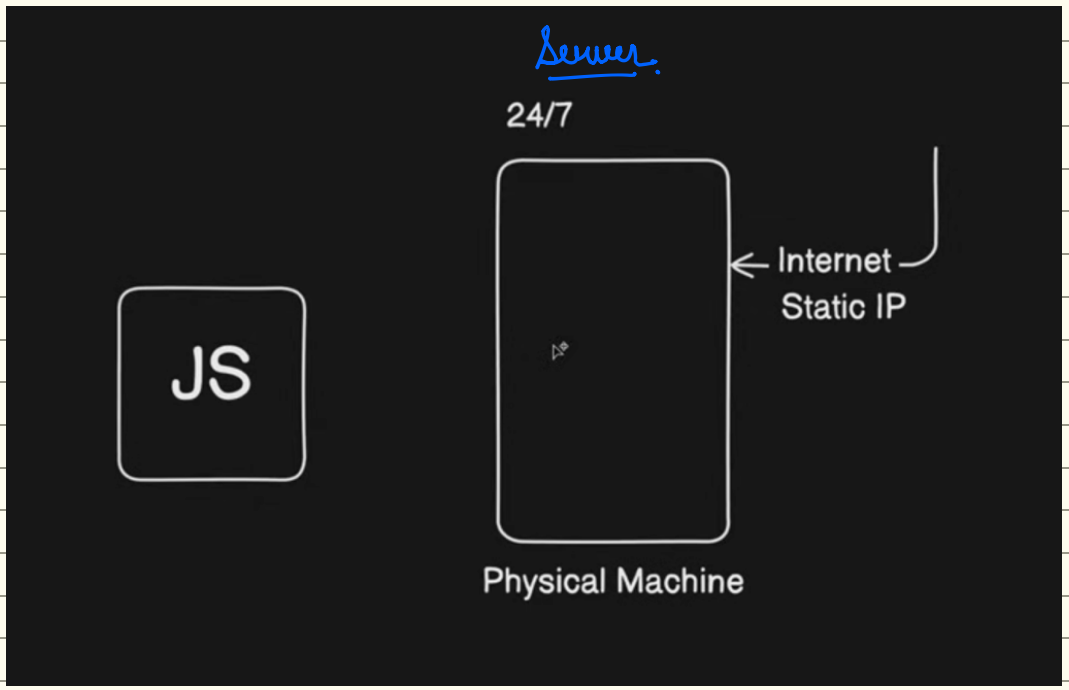
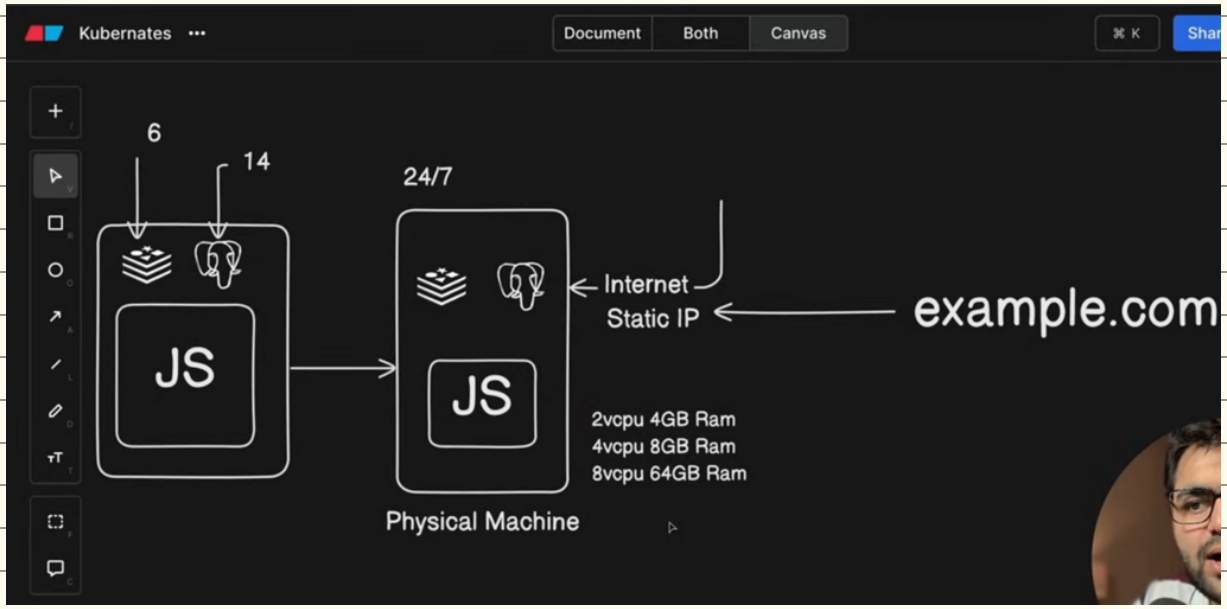
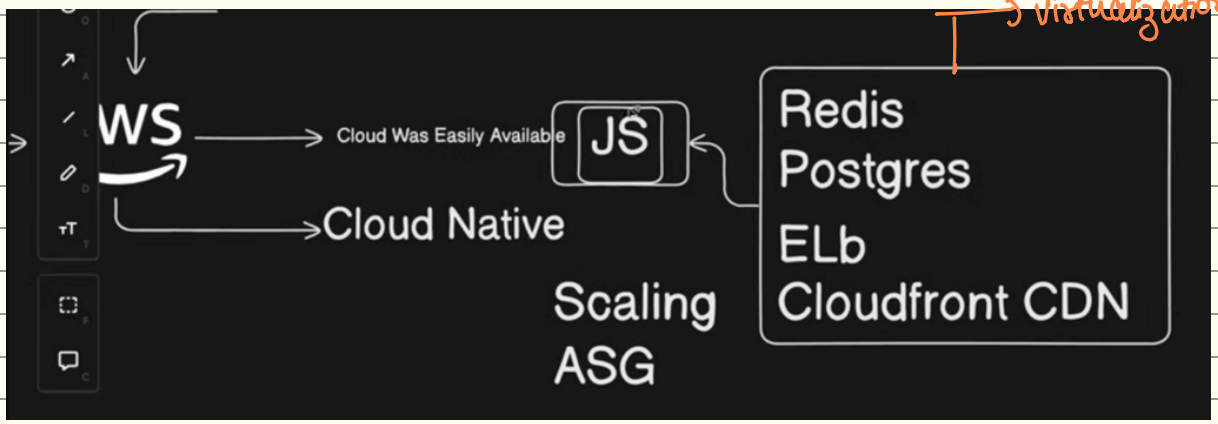
KUBERNATES

↳ Explained !!!

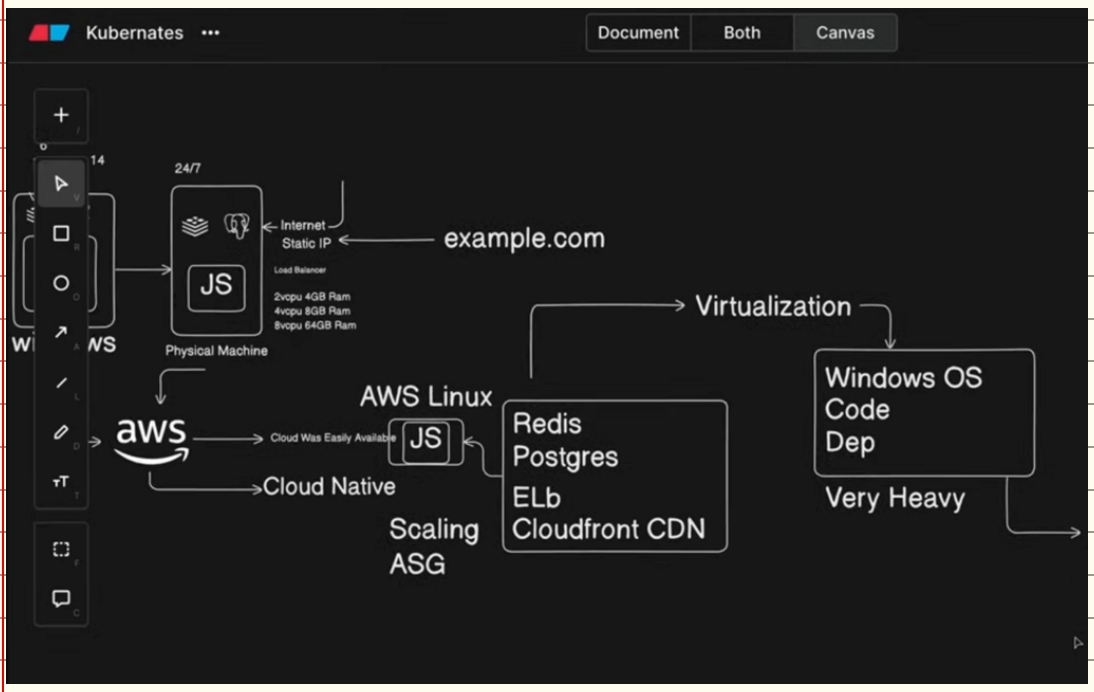




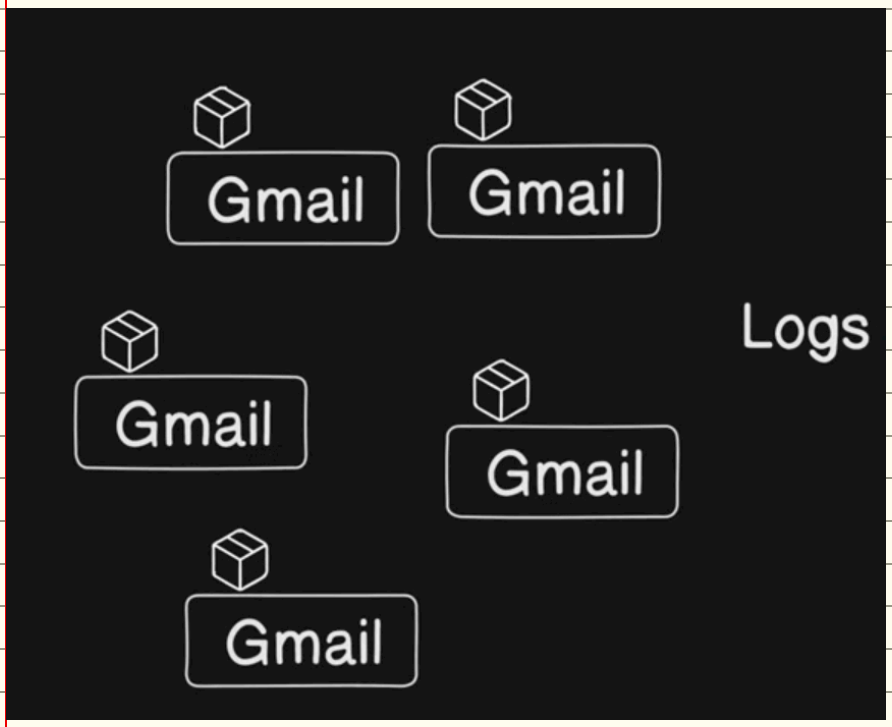
After the emergence of AWS :-



Common Issue: It works on my computer., Hence people started migrating to cloud Native tech, where virtualization is eg.



Very Light weight Containers.



Maintaining Logs, managing, upgrading, scaling containers is a very difficult task.

↓

& hence we Orchestrate Containers.

→ Containers Orchestration:-

Container orchestration is the process of automating the management of containerized applications, including their deployment, scaling, and security. It helps organizations to take advantage of containerization without incurring extra maintenance costs.

∵ In the early era of Containerization.

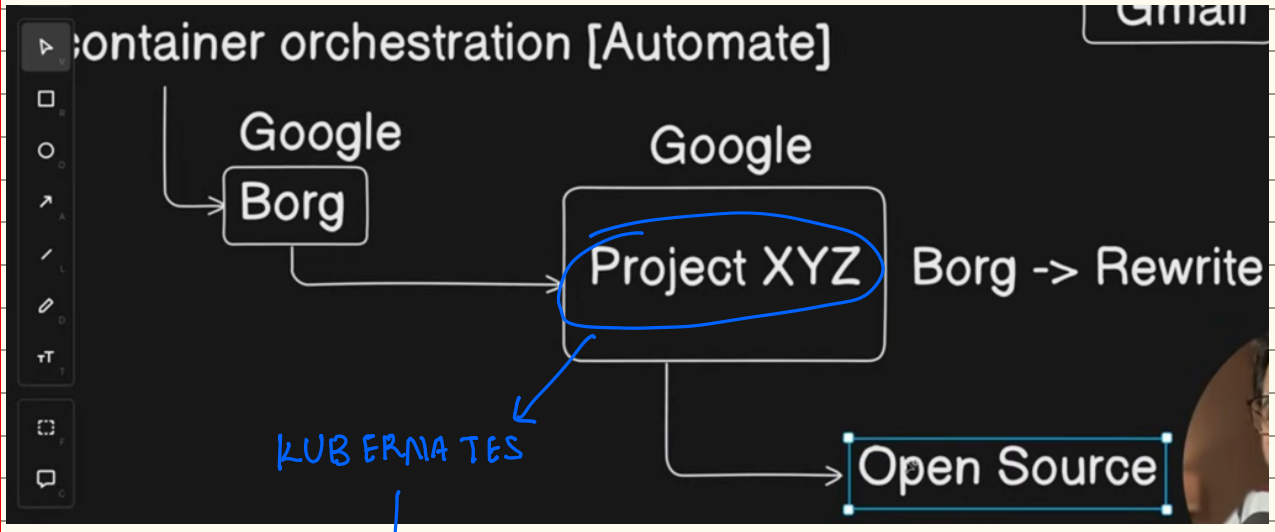
↳ google was one of the early staged company, adopting to container orchestration.

∵ It introduced Borg., for internal containers management

Google Research
http://research.google › pubs › large-scale-cluster-manag...

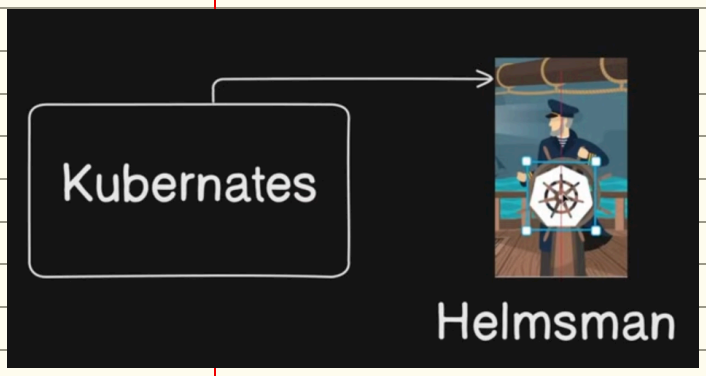
[Large-scale cluster management at {Google} with {Borg}]

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of ...



later donated to CNCF
↓
Cloud Native Computing Foundation.

→ Kubernetes:-



Kubernetes (often abbreviated as **K8s**) is an **open-source container orchestration platform** used to **automate:**

- ✓ Deployment of containerized applications
- ✓ Scaling (adding/removing instances automatically)
- ✓ Load balancing
- ✓ Resource management
- ✓ Rolling updates and rollbacks
- ✓ Self-healing (restart/replace failed containers)

Why do we need Kubernetes?

When you have **many applications** running in containers (like Docker), managing them manually gets complex:

Without Kubernetes	With Kubernetes
Manual deployment of containers	Automated deployment and scaling
Hard to recover failures	Self-healing automatically
Hard to distribute load	Built-in load balancing

Kubernetes, also known as K8s, is an open source system for automating deployment, scaling, and management of containerized applications.

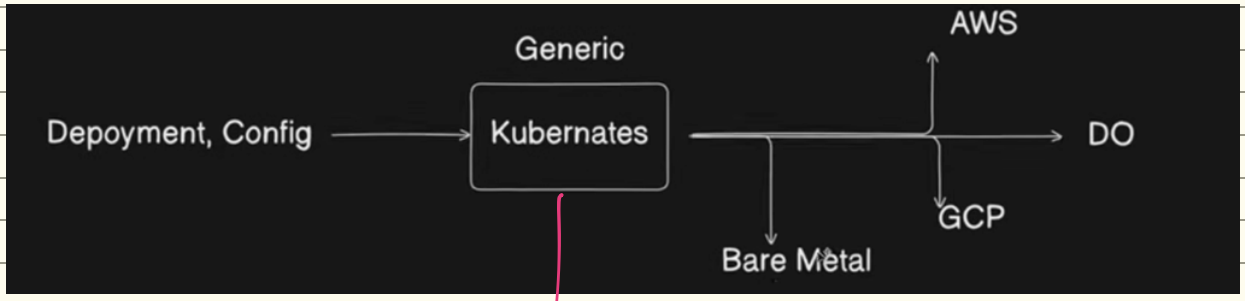
It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

Key Concepts

Term	Meaning
Cluster	A group of machines (nodes) managed by K8s
Node	A worker machine (VM or physical server)
Pod	Smallest deployable unit → one or more containers
Deployment	Blueprint for how many pods, what version, scaling rules
Service	Stable network endpoint to access pods
Ingress	Routes external traffic to internal services

Example Use Cases

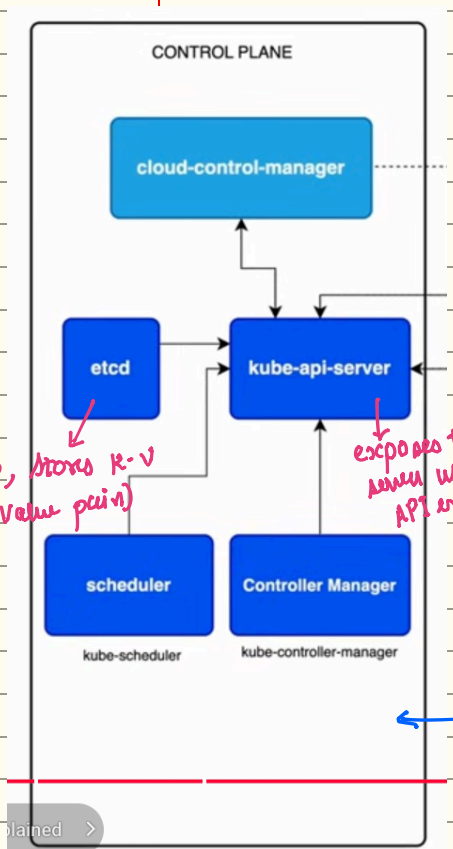
- Microservices applications
- Scalable web platforms (Netflix, Spotify use it)
- CI/CD pipelines
- Edge / hybrid cloud infrastructure



Helps to develop generic configuration for deployment on all cloud native platforms, hence prevents vendor locking.

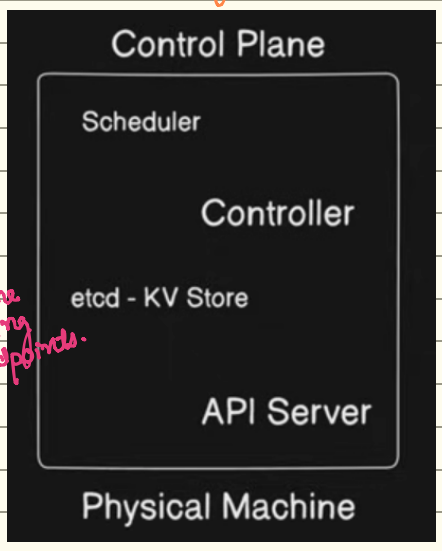
→ Architecture for Kubernates :-

- physical machine is the control plane.
- In this you need :-
  (i). scheduler
  (ii). controller
  (iii). etcd - KV store
  (iv). API server.



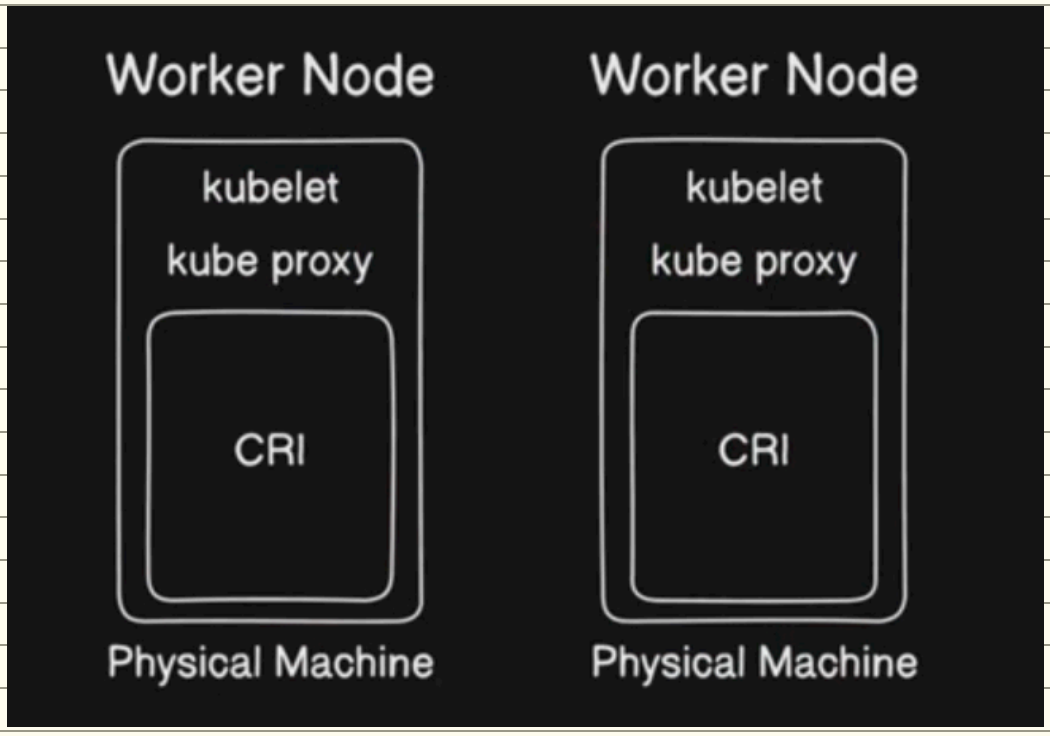
DB, stores K-V (Key value pair)

exposes the server using API endpoints.

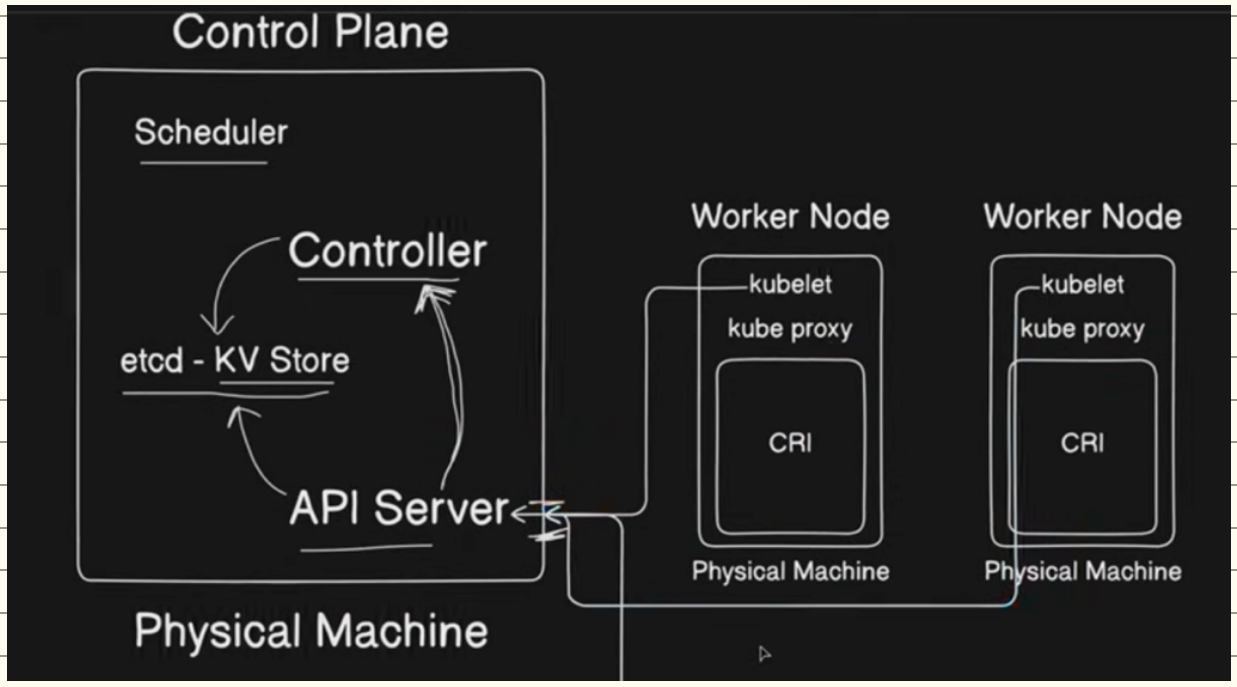


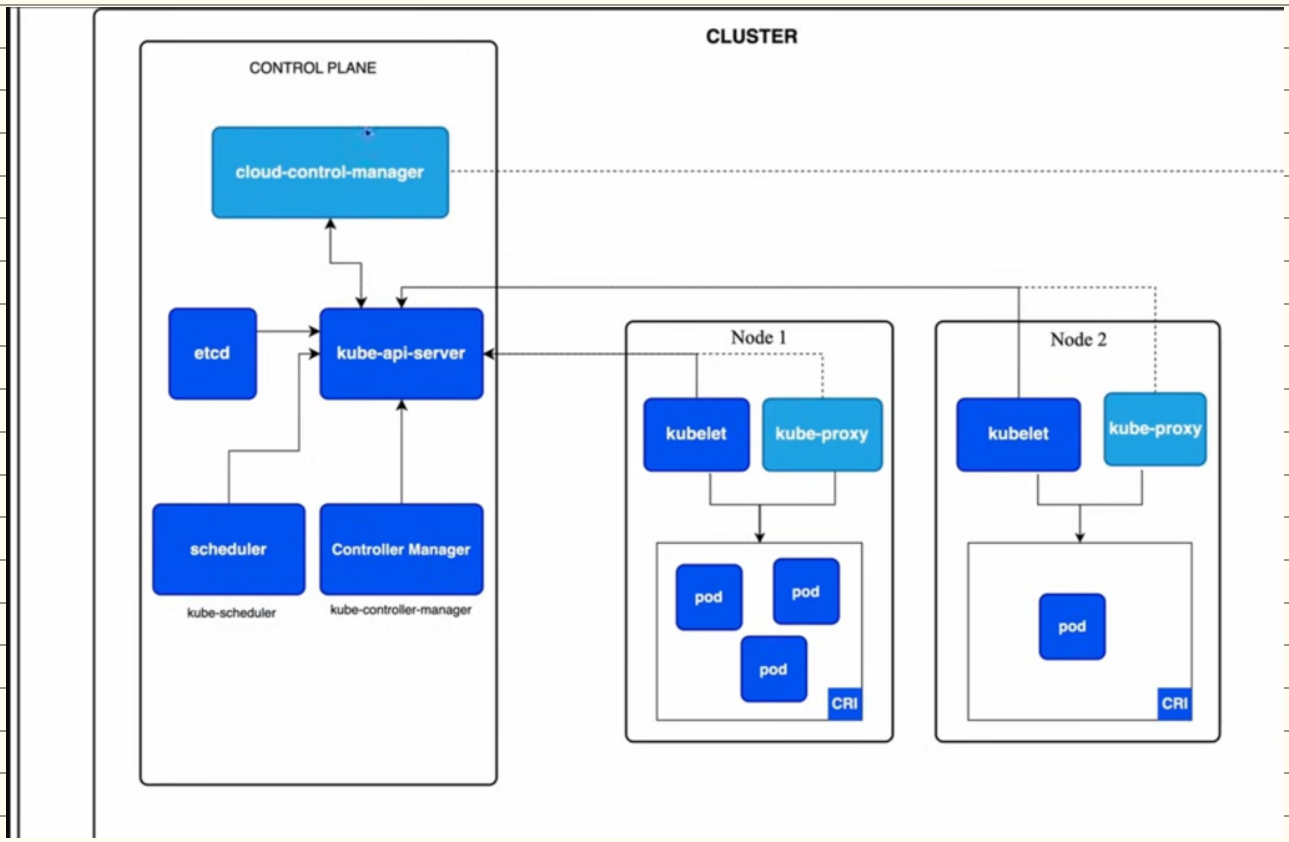
Deployment, Config.

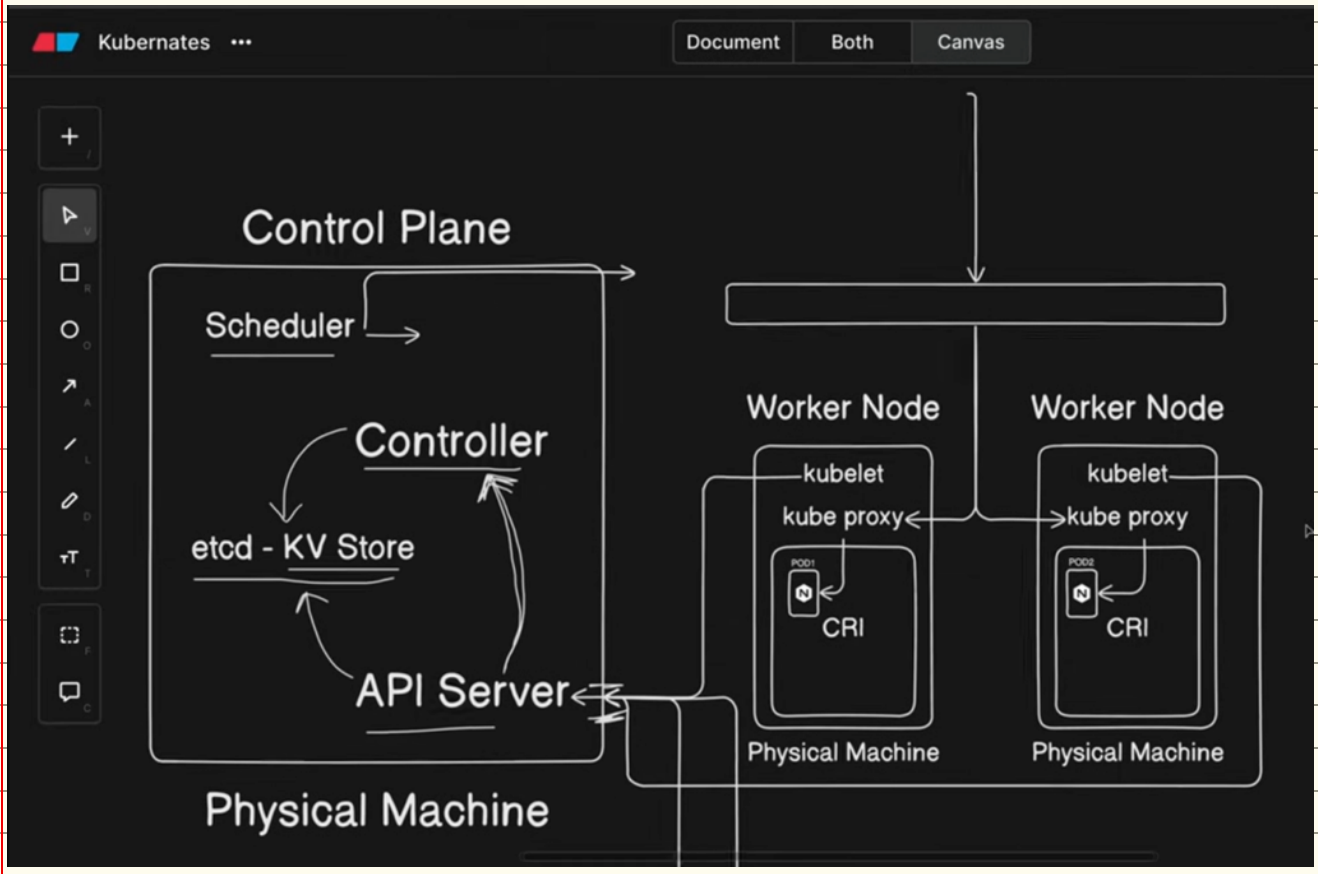
- Worker Node are different physical machines, where the work load runs.
  ↳ Here the actual (containers) applications run on this worker nodes.



- The kubelets in the worker nodes are connected to the API server.







• kube proxy provides:- N etworking and Traffic rules to the kublete.

→ Container Runtimes :-

Container Runtimes

1. Podman
2. CRI-O
3. Containerd (CNCF)
4. LXC (Linux Containers)
5. Kata Containers
6. Firecracker (AWS Lambda)

→ Working of Kubernetes :-

🚀 Kubernetes Cluster — High-Level Working

A **Kubernetes cluster** consists of:

1️⃣ Control Plane (Brain of the system)

Manages **what should run** and **where.**

2️⃣ Worker Nodes (Muscles of the system)

These actually **run the application** containers.

🧠 Control Plane Components

Component	Responsibility
API Server	Entry point — all communication happens through this (kubectl → API Server → Cluster)
Scheduler	Decides **which node** will run each Pod based on resources
Controller Manager	Ensures the desired state: if a pod crashes → recreate
etcd	Key-value store — keeps the cluster state (like a database for configs)

💪 Worker Node Components

Component	Responsibility
Kubelet	Talks to the API Server, starts/stops containers on that node
Container Runtime	Runs containers (Docker, containerd, CRI-O)
Kube-proxy	Handles networking & load balancing to Pods

📌 How Kubernetes Works (Step-by-Step)

Example: You tell Kubernetes to run 5 instances of your app.

1. 👷 You submit a Deployment YAML → `kubectl apply -f app.yaml`
2. 📩 API Server receives request → stores desired state in **etcd**
3. 🧠 Scheduler checks all nodes → picks the best nodes for 5 Pods
4. 🤝 Kubelet on selected nodes runs containers via container runtime
5. 🔁 Controller Manager keeps checking:
   - If any Pod crashes → recreate automatically
   - If demand increases → scale up
   - If node dies → move Pods to another node
6. 🌐 kube-proxy ensures networking between Pods and Services

🌍 Cluster Working Visualization

lua

Copy code