Kubernetes :-

* Managing Containers.

Monolithic Applications :- Application running in Single Container.

Micro Services :-

* Each functionality deployed differently.

* fault Isolation.

Orchestrators :- Deploying & managing Containers Dynamically

Control Plane → Control Plane + Nodes

kubectl → CLI tool

Pod → Scheduling Unit (Containers are inside)

---

① Create Micro Service

② Containerize it

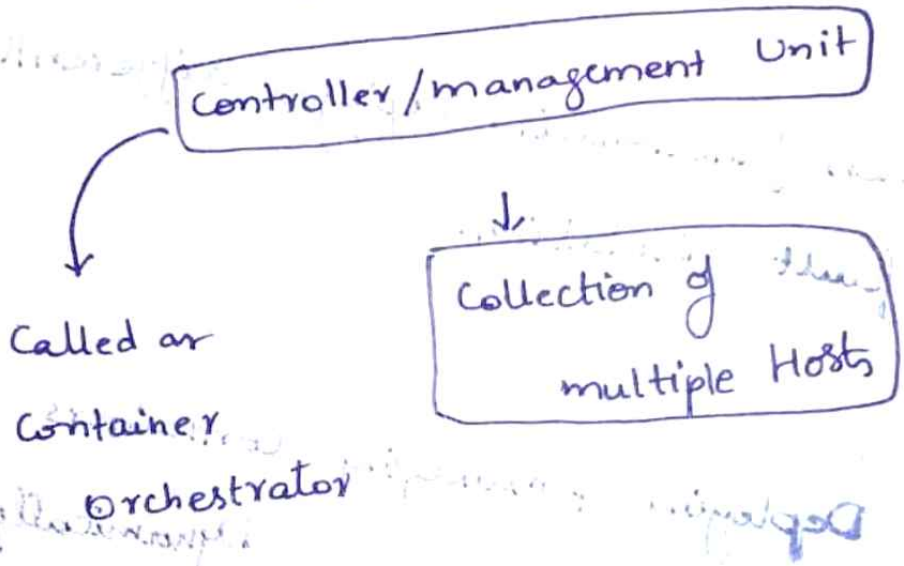
③ Put Container in Pods.

④ Deploy

fault tolerant & scalable

↓

achieved by building



Why Orchestrators ?

we can manually maintain a couple of containers, write scripts for dozens of containers.

But,

Orchestrator makes things much easier for user when it comes to handling hundreds & thousands of Containers.

(Monitoring Containers)

(Scaling)

(Networking)

(Self Healing)

Kubernetes from 40K feet :-

* Cluster to run applications
* Orchestrator of cloud-native micro services app.

→ Bunch of machines to host applications.

Kubernetes cluster → control plane → Brain of k8s
& worker nodes → Muscles of k8s

Orchestrator → fancy word for managing & deploying applications.

---

* How to run applications? (on kubernetes)

1. Design the application as small independent microservices
2. Package each service as its container
3. Wrap each container in kubernetes Pod
4. Deploy pods to cluster via higher level controllers such as Deployments, DaemonSets, StatefulSets

The kubernetes Control Plane :-

Server running collection of System Services.

* API Server (all roads lead to API Server)

Internal System components & external components, all communicate via API Server

---

* Cluster Store (only stateful part of the control plane)

(no cluster store, no cluster)

Persistently stores entire config & state of cluster

---

* The Controller Manager & Controllers → moniter cluster components & respond to events.

↳ Controller of controllers

Eg:- Deployment Controller, StatefulSet Controller, etc

Single Aim of Controller :-

Ensure observed state of cluster matches the desired state.

1. Obtain desired State
2. Observe current State
3. Determine Differences
4. Reconcile Differences.

The Scheduler :-

(Not responsible for Running tasks,
just picking nodes to run them)
(Kubelet runs the tasks)

The cloud Controller Manager :-

facilitate integration with cloud Server.

Worker Nodes

* Watch API Server for new work Assignments
* Execute Work assignments
* Report Back to the control Plane.

Kubelet :- (Daemon)

*main kubernetes agent & runs on every node

*watch the API server for tasks, execute the task if it could,
report the status back

Container Runtime :-

*Kubelet needs container runtime for container related tasks.

*In early days, kubernetes had native support for Docker.

*Recently moved to plugin model called Container Runtime Interface (CRI)

↓

Supports various runtimes other than Docker.

Kube Proxy :-

Responsible for local cluster networking.

Declarative Model & desired state :-

→ (Heart of kubernetes)

Declare a desired state & strive to maintain it at all costs.

Pod :-

Kubernetes demands that every container runs inside a pod.

k8s Pod is a construct for running one or more Containers

The Simplest model is to run a single container in every Pod. However, there are some cases that need to run multiple containers in a single Pod.

If you need to scale an app, you add or remove pods. You do not scale by adding additional containers inside the Pod.

~~Single~~ Pod → Single Host → Pod resides on a Single Host

Updating Pods :-

Delete old one & replace with new One.

Pod → mortal,
atomic & immutable.

Working with Pods :-

↳ Atomic Unit of Deployment in k8s.

Defined & Deployed using YAML manifest files.

Normal to deploy Pods with high level controllers like Deployment & Daemonsets.

Static Pod → Not deployed with high level controller
↳ Doesn't have advantages like self heal provided by controllers.

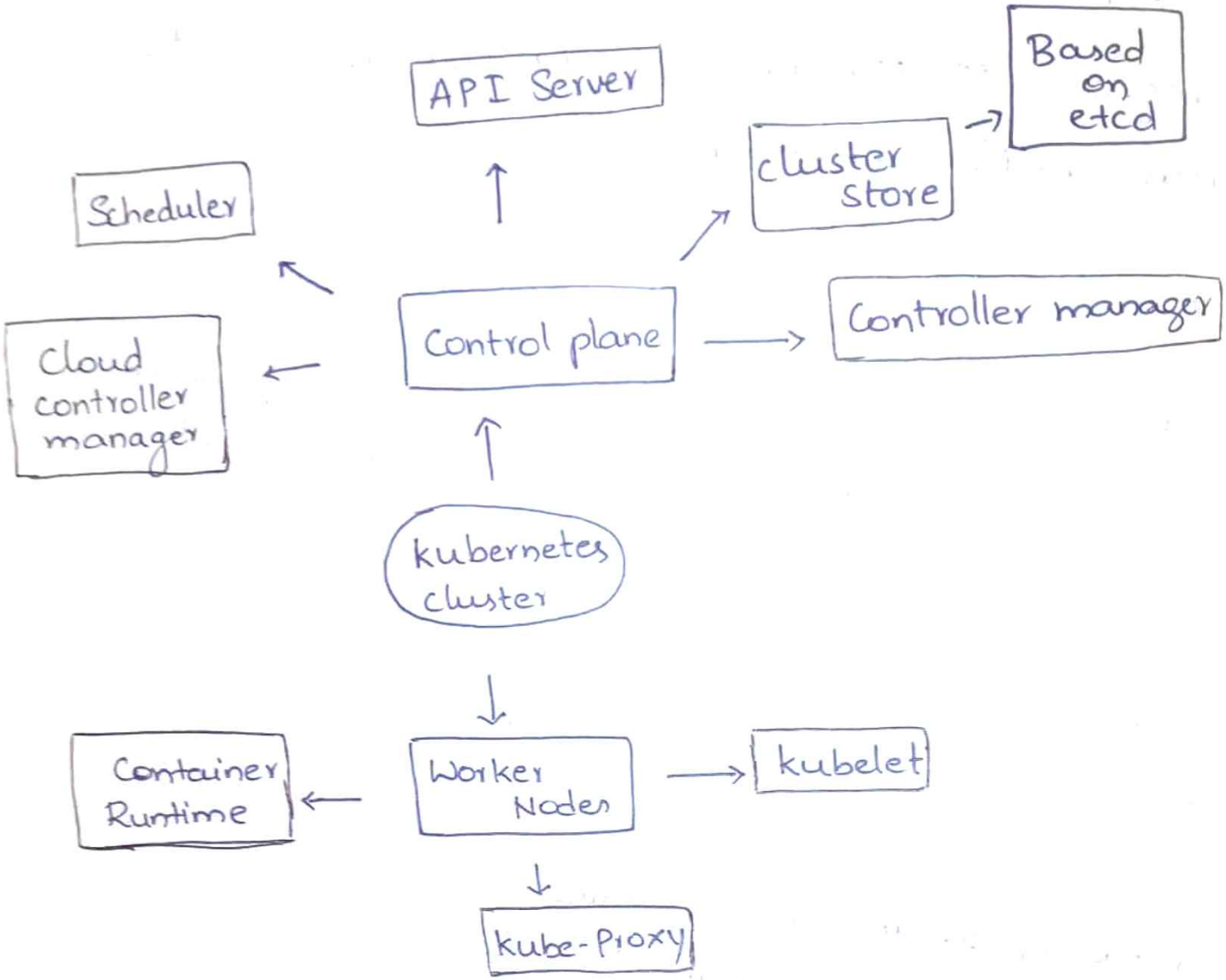
Kubernetes Services :- (Pods are mortal & unreliable, should not connect directly to them)

(provides Stable Networking for Pods)

* Every Service own STABLE IP Address
  own STABLE DNS name
  own STABLE Port.

* Services use "labels" & "selectors" to dynamically select the pods to send traffic to.

* Service is Observing changes & updating its list of healthy Pods. But never changes IP, DNS or port.


API Server
Based on etcd
cluster store
Scheduler
Control plane
Controller manager
Cloud controller manager
kubernetes cluster
Container Runtime
Worker Nodes
kubelet
kube-Proxy

kubectl :-
main kubernetes command line tool.

kubectl converts user friendly commands to HTTP Rest requests with JSON content required by the API Server.

kubectl configuration file → "kubeconfig"
↓
It contains definition for
- "Clusters"
- "Users"
- "Contexts".