KUBERNETES

Date

→ Open source, container orchestration tool
→ Helps you manage containerized applications in different environment

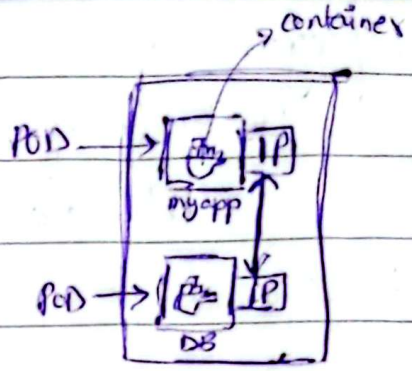
- The need for a container orchestration tool
  → Trend from Monolithic to microservice architecture
  → Increase usage of containers
  → Demand for a proper way of managing those hundreds of containers

- Features
  ↳ High availability / No downtime
  ↳ Scalability or high performance
  ↳ Disaster recovery - backup and restore

→ Through Kubernetes we can run our app on any cloud provider. It provides an abstraction layer for us.

Date

▷ Main Components of Kubernetes



- Pod
  - ↳ Smallest unit of K8s
  - ↳ Abstraction over container
  - ↳ It contains one or more application containers that are tightly coupled
  - ↳ But mostly we should run one-container per Pod
  - ↳ It creates a layer of running env on top of container
    - ↳ K8s has made this abstraction so that we are not dependent on ~~contai~~ a specific containerization technology.
  - ↳ Each Pod gets it's own IP Addr (not the container!)
    - ↳ Pods can communicate ~~with~~ with each other using IP
  - ↳ Pods can die easily, so when they created again they get new IP. So if two Pods communicate each other then we should update the IP.

- Service
  - ↳ For this reason service is used
  - ↳ It is basically a permanent IP Addr that can be attached to each Pod-

→ Service is also a load balancer for Pod. Like we have two Pods of AddToCart Service. So the service sends request to the free Pod.

Date

↳ Lifecycle of Pod & Service not connected.
↳ Even if Pod dies it will not loose it's permanent IP Add.

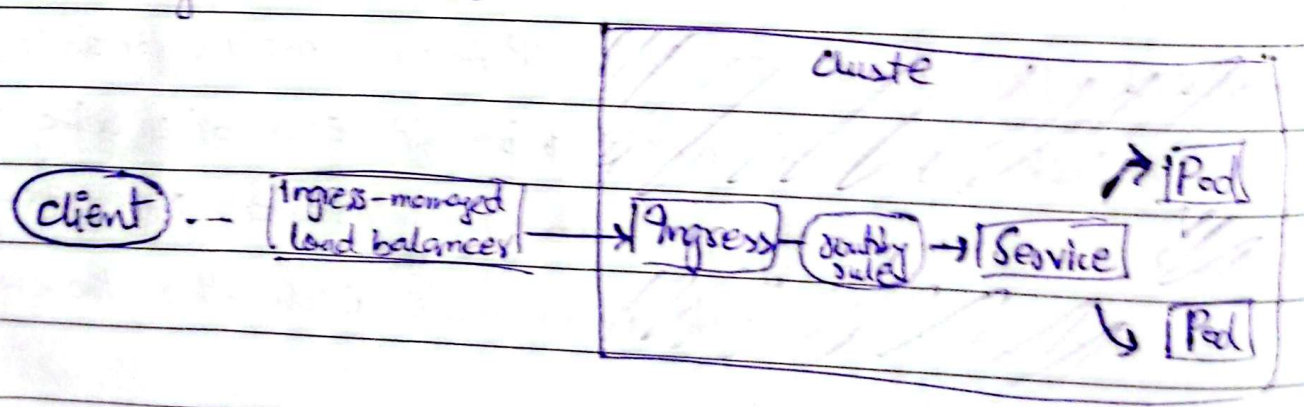
→ External Service (like we need to run an app on browser (easily accessible))

→ Internal Service (like DBs)

- Ingress

→ Exposes HTTP & HTTPS routes from outside the cluster to services within the cluster.

→ Traffic routing is controlled by rules defined on the ingress resource.



Date

- ConfigMap
  - ↳ Used to store non-confidential data in key value pairs
  - ↳ Pods can use them as env variables.
  - ↳ ConfigMap allows you to decouple env-specific configuration from your containes images
  - ↳ ConfigMap can also be mounted as data volumes

- Secret
  - ↳ It contains small amount of sensitive data such as a password / a token or a key.
  - ↳ they are similar to ConfigMap.
  - ↳ they store data in base64 encoding format

- Volumes
  - ↳ By default there is no data persistence on restarting of Pod.
  - ↳ But can be done ~~as~~ thru volume
  - ↳ This volume is seperate from cluster.
  - ↳ It can be the part of the local machine or remote

ReplicaSet: is a Kubernetes controller that ensures a specified no. of identical Pods are always running

↳ Pod Blueprint

Date

- Deployment & Statefulness

↳ A deployment manages a set of Pods to run an application workload

↳ Usecase of Deployment

↳ It helps creating new replicas of existing pods

↳ If we change a pod then it helps to scale up the new one & gradually scale down the older

↳ If a pod crashes, it rolls back to stable state

↳ Helps in auto scaling (balances load automatically)

↳ Cleans up the older state

→ Deployment is an abstraction over Pods.

→ We work with Deployment

→ We cannot replicate DBs using Deployment because it would cause data inconsistencies. (Call DB pods sharing same Database)

↳ This can be done thru StatefullSets

↳ It makes sure that DB read/writes are sychronized

↳ But it is not easy to work with Statefullsets.

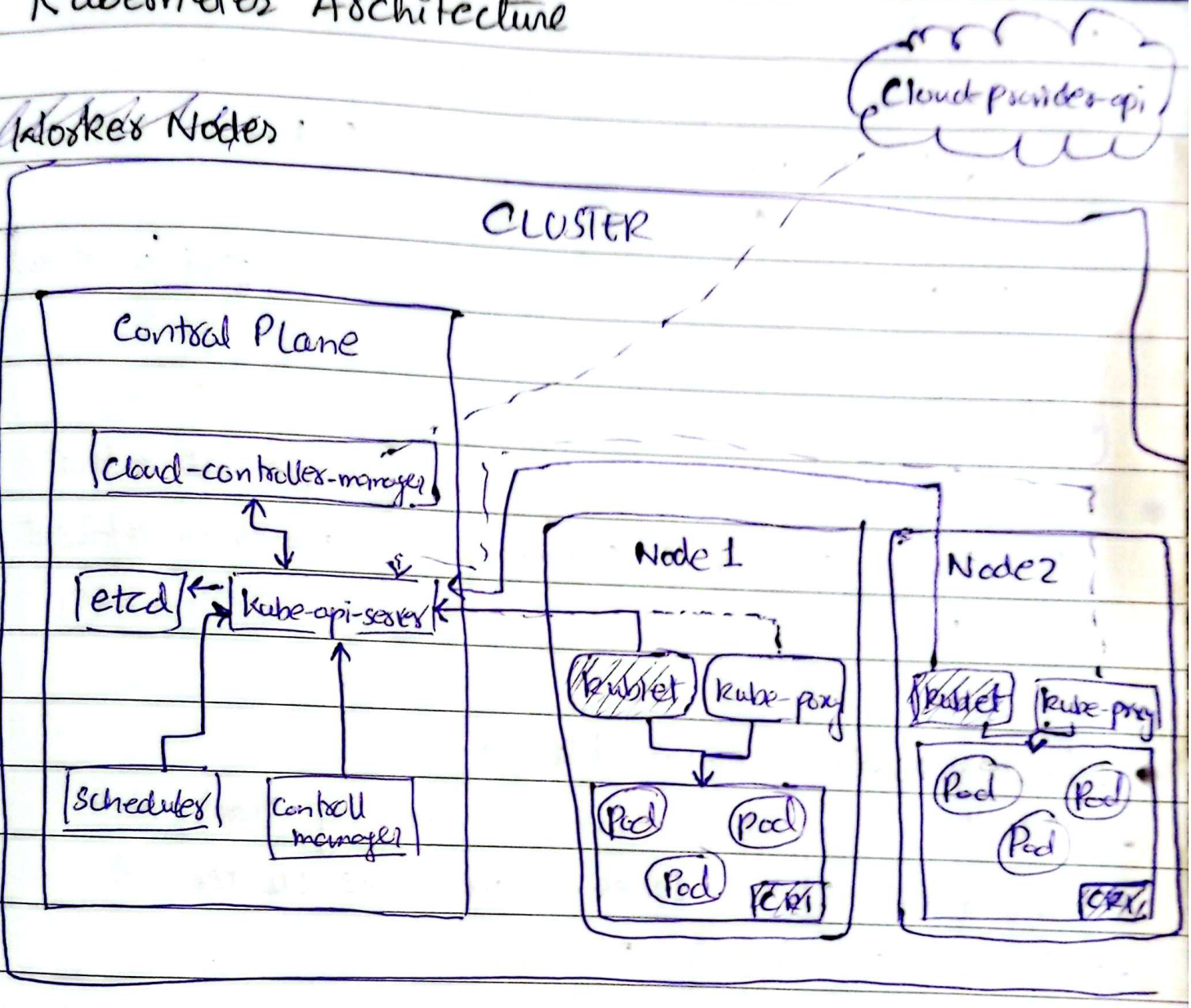
↳ recommended to use Statefull apps outside K8S cluster

Page No.

Date

• Kubernetes Architecture

• Klorker Nodes



• klorker Node

↳ One of the main component

↳ Each node have multiple Pods running on it.

↳ Each node have 3 process

(1) Container Runtime
(Docker Engine)

Page No.

Date

② Kublet
↳ This is the process that schedules the pods
↳ It is the interface to both CRI and the machine (node)
↳ It assign resources from node to pod (containers)

③ Kube Proxy
↳ It has intelligent forwarding logic inside that makes sure that communication works in efficient way.

• Master Node (Control Plane)
→ All the Managing process are done by Master Node
→ It makes global decesion about the clusters
→ Process running in Master Node
① Api-Server
↳ It is like a cluster gateway
↳ It is the frontend for the Kubernetes control plane
↳ The client interacts with Api-server.
↳ It gets the initial updates
↳ Act as a gatekeeper for authentication

Page No.

Date

(2) Scheduler

↳ If we need to add a new pod Api-server sends req to Schedules.

↳ It intelligently identifies on which node to create the Pod

↳ Schedules on least busy Node.

↳ Schedules only decide on which node to create

↳ the actual creation is done on Node by Kublet

(3) Controller Manager

↳ If a pod die on a Node so to manage & reschedule it it is done by this process

↳ Detects cluster state changes

↳ It request the schedules to create new pods

(4) Etcd

↳ Key value store of cluster

↳ Used as Kubernetes backing store for all cluster data

↳ Etcd is the cluster brain

↳ Application data is not stored (only cluster state info is stored)

Date

→ Master Node is replicated and Api-server is load balanced.

→ Master Nodes req less resources than Worker Nodes