KUBERNETES

The process of automatically deploying & managing containers

Open Source container Orchestration Tool

Developed by Google

Kubernetes

Helps you manage containerized applications in different deployment environments

- Physical
- Virtual
- Cloud
- Hybrid

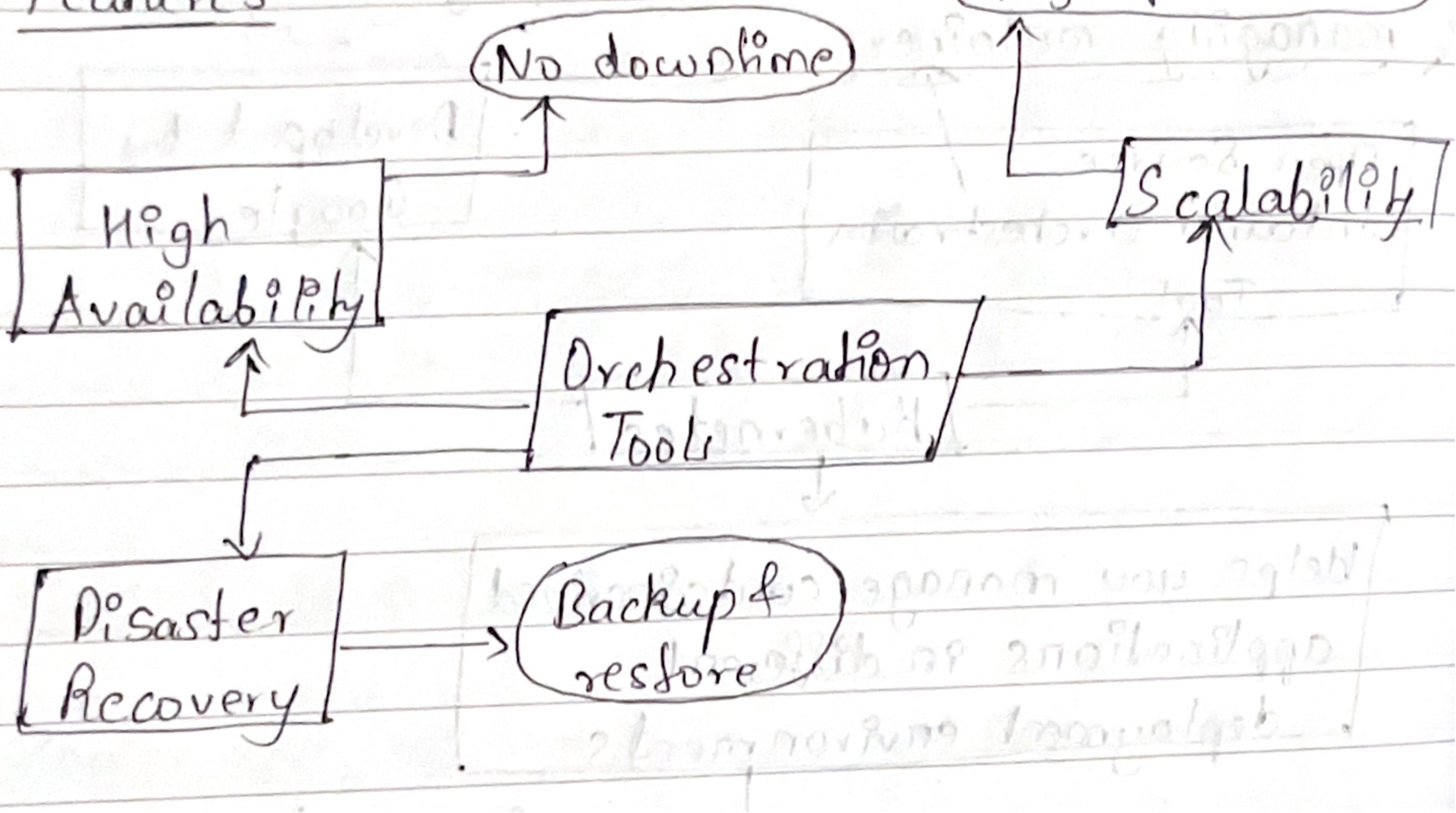
Need?

Trend from Monolith to microservices

Increased usage of containers

Demand for proper way for managing those hundreds of containers

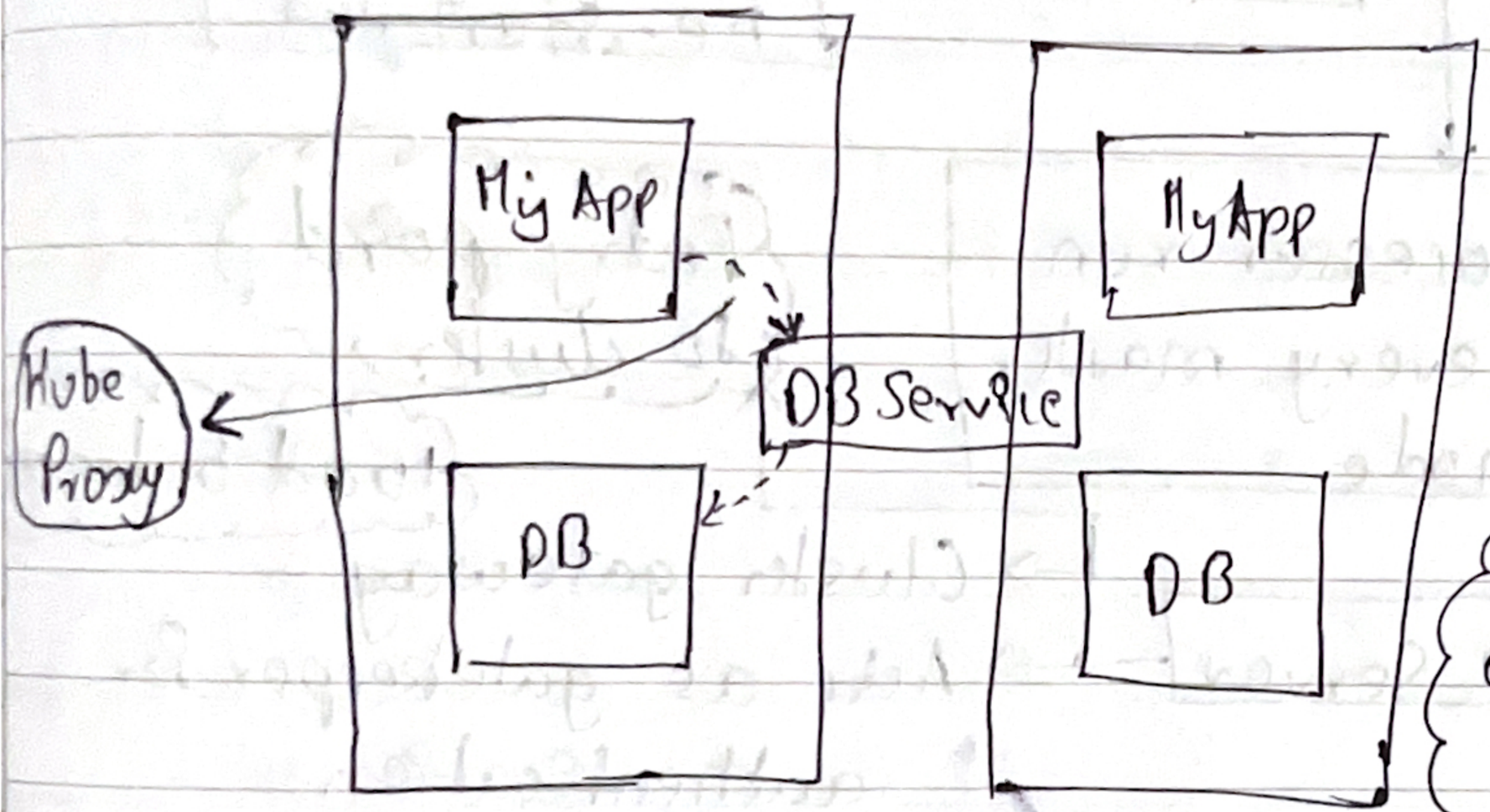
Features


No downtime
High performance
High Availability
Scalability
Orchestration Tools
Disaster Recovery
Backup & restore

· Kubernetes Architecture

Worker machine in K8 cluster.

Collection of Nodes



Each Node has multiple Pods

A machine, physical or virtual on which K8s is installed

3 Processes must be installed on every node

Workers nodes do the actual work

Node processes

→ Container runtime [ex: docker]

→ Kubelet → Process of Kubernetes

→ Interacts with both container & node

→ Starts the pod with container inside Pod

→ Also assigns resources to container from node

→ Kube Proxy → Forwards the requests

Managing Processes

Master Processes

- Scheduler Pod
- Monitor
- Re-schedule/ Re-start pod
- 4 Processer run on every master node

Entry point to cluster

Load balanced

- Api Server
  - → Cluster gateway
  - → Acts as gatekeeper for authentication
- Scheduler
  - → Schedules new Pod
  - → Decides on which node new Pod should be scheduled
    - → Based on available resources on Pods
    - Kubelet starts the pod after getting request from scheduler
- Controller Manager
  - ↓
  - Detects cluster state change
    - → Ex: crashing of pods
    - → Tries to recover
- Etcd
  - → Distributed storage across all master nodes
  - → Cluster brain
  - → cluster changes get stored in key value store

Is cluster healthy?
Resources are available
Did cluster state change?

Note:
Application data is not stored in etcd

Client request → API server → validates request → other process... → Pod

Schedule new Pod → API server → Scheduler → Where to put pod? → Kubelet

Controller Manag → Scheduler → Kubelet → Pod

Pod present in etcd

MiniKube & Kubectl

Creates virtual box on laptop → Nodes runs on that Virtual box

runs ← MiniKube → 1 Node K8s cluster

MiniKube → Both master & worker processes in one Node

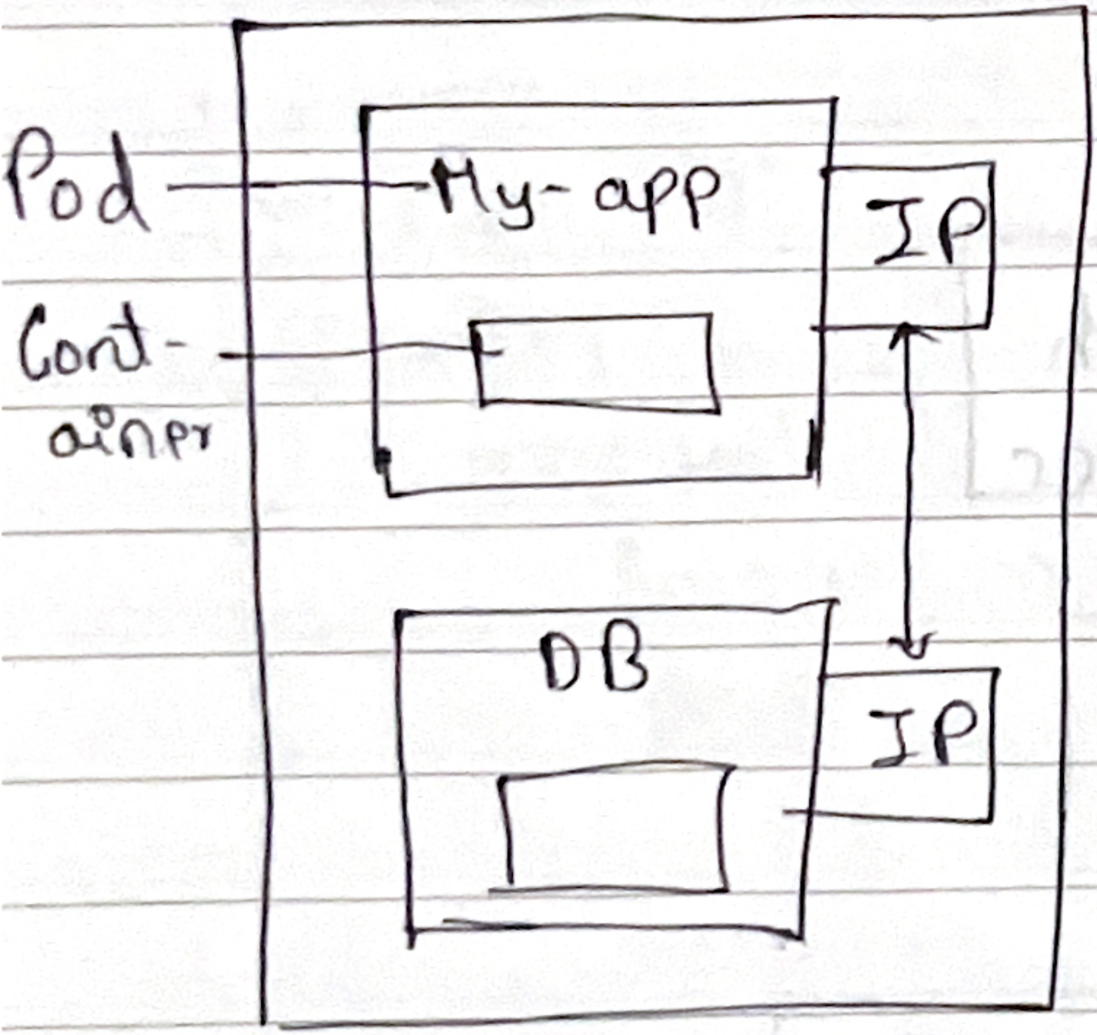
MiniKube → Testing purposes

Kubectl → For interaction with mini cluster

Kubectl → Command line tool for K8s cluster

Kubernetes Components

i) Node & Pod



Pod

→ Smallest unit of K8s

→ Abstraction over container

→ Usually 1 application per pod

→ Each pod gets its own IP address

→ New IP address on re-creation

Solution → Services

* If Pod dies, then new pod will be created with new IP address which is problem in maintaing IP address

ii) Service & Ingress

Service
- → Permanent IP address
- → Lifecycle of Pod & services Not connected

Ingress ⟶ Routes traffic into cluster
- → Manager routing rules
  - → Manage external users access to service in Kubernetes cluster

iii) Config & Secret

Don't put credendiats into config Map

Config Map
- → External configuration of application
  ex: DB_URL = mongo-db

Secret
- → Used to store secret data
  - → base64 encoded

→ Use it as environment variables or as properties file

iv) Volumes

Volumes

↳ Storage on local machine
or Remote, outside K8s cluster
↳ Cloud

K8s doesn't manage data persistance

v) Deployment & StatefulSet

Deployment → used for stateless Apps

→ Blueprint for pods i.e, duplicate pods will be created with connection to same service [Availability]

DB can't be replicated - via deployment

→ Create deployments

→ Abstraction of pods

StatefulSet

↳ used for stateful apps [Eg: MongoDB, MySQL]

* DB are often hosted outside of K8s cluster

Note:
Kubectl is tool used to interact with any type of K8s cluster

Kubectl commands

-> kubectl get pod ⟶ to get pods

-> kubectl get services

*Pods are smallest unit of Kubernetes. To create pods you have to create deployments. Deployment is the abstraction over Pods

-> kubectl create deployment NAME --image=image
ex:- kubectl create deployment nginx-depl --image=ngin

-> kubectl get deployment -> kubectl get pods -o wide
-> kubectl get replicaset
⟶ replicaset is managing the replica of pod

-> kubectl logs <pod-name>
-> Kubectl describe pod <pod_name>
-> kubectl exec -it <pod_name> --bin/bash
↳ get interactive terminal of of container
-> kubectl delete deployment <deployment_name>
-> kubectl apply -f <config_file.yaml>

Service configuration file

Ex:

apiVersion : v1
kind : Service ⟶ Service
metadata:
  name: mongodB-service ⟶ random Name
spec:
  selector: ⟶ to connect to Pod through label
    app: mongodb
  ports: ⟹
  - protocol: TCP
    port: 27017 ⟶ service ports
    targetPort: 27017 ⟶ containerPort of deployment

Secret config file

Ex,

apiVersion : v1
kind : Secret ⟶ Secret
metadata: ⟶ random name
  name: mongodb-secret
type: Opaque ⟶ 'Opaque' - default for arbitrary key-value pairs
data: ⟶ actual contents in key-value pairs
  username:
  password:

Note: username & password should be in base64 format