What is Docker? →

A platform for — Building
Running &
Shipping Applican's.

Application developed on one machine ⟶ can Run & developed on other machines.

Reasons — i) One or more files missing.
ii) Software version mismatch.
iii) Diff configuration settings
(env variables, etc)

Package
↓
Our Applican
↓

Ex. Node 14, mongo 4 APP → Docker ↓ $docker-compose up → Runs Downloads & Runs the dependencies
↓
inside isolated env of a container.

Delete Applican with all its dependencies in one go -

```
$ docker - compose down --rmi all
```

Virtual Machines Vs Container.

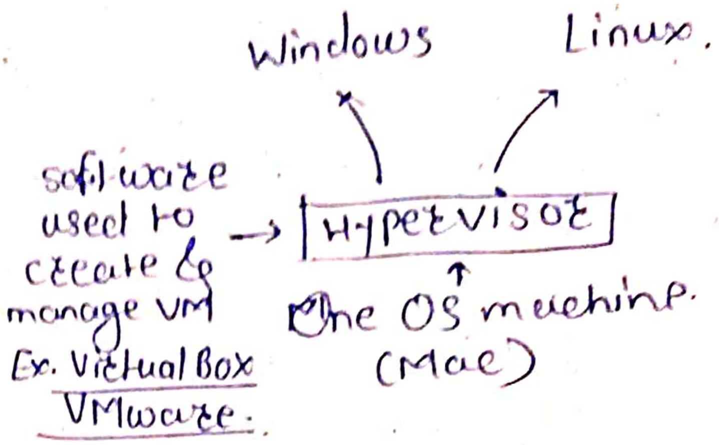
Container

An isolated environment for running an application.

- Allow running multiple apps in isolaⁿ.
- Are light weight.
- Use OS of the host (don't contain full OS of own)
- Start quickly.
- Needs less hardware resources.

Virtual machine.

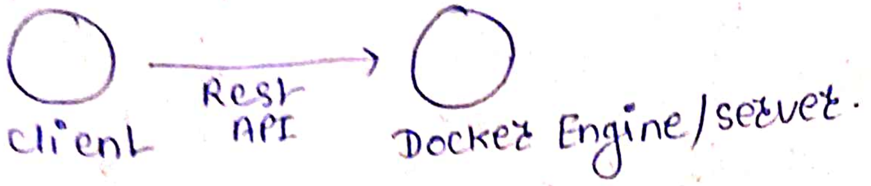
An abstraction of a machine (physical Hardware)



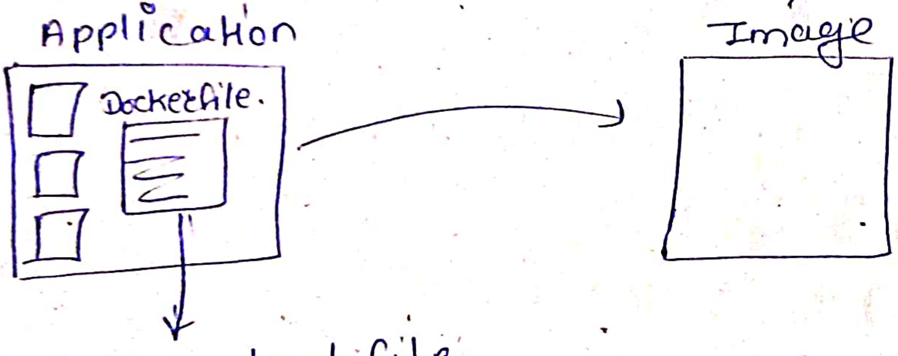
Problems -

- Each VM needs a full-blown OS.
- Slow to start.
- Resource Intensive.

Docker Architecture—



Docker Workflow.

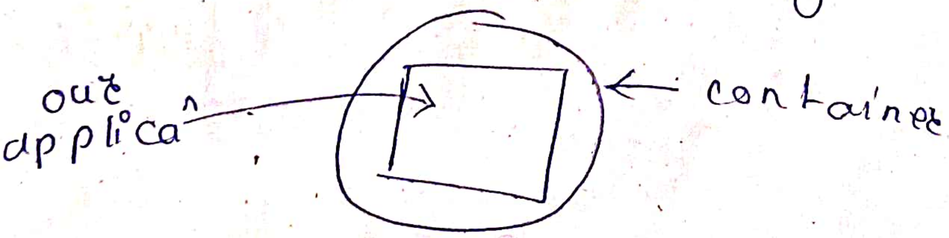


has every thing our appⁿ needs to run

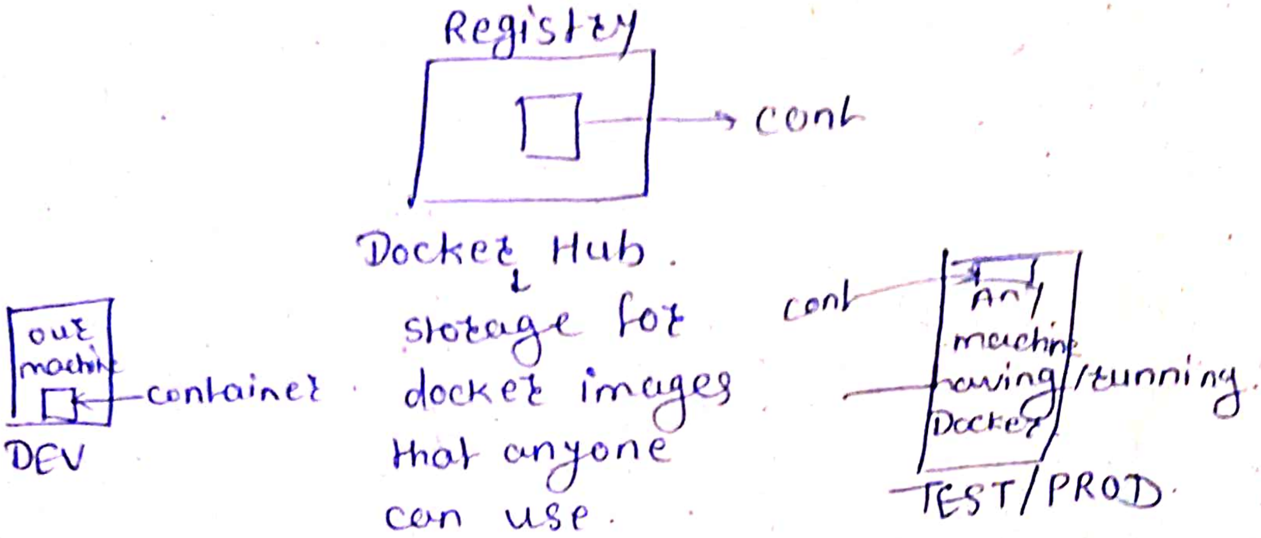
i) Cut-down OS
ii) A runtime env (eg. Node)
iii) Applicaⁿ files.
iv) Third party libraries
v) Env. variables.

Plain text file with docker usage → to package this applicaⁿ into an image.

Container — special kind of process having its own kind of file sys. which is provided by an image.



So insted of running an applicaⁿ in perticular Process
↓
We tell docker to run it inside a container (an isolated env)



Docker Hub.
↓
storage for docker images that anyone can use.

Docker in Action →

start with an Os

Install Node

Copy app files.

Run node app.js

First project

Docker file → ~~com~~

```
FROM node:alpine
COPY . /app
WORKDIR /app
CMD node ~~app/~~ app.js
```

install → node

(for small applica linux ~~structure~~ Distribution. Ex. Ubuntu fedora Debian sentos Alpine. → alpine

. → all files

/app → to this directory.

CMD → instrucⁿ to execute comma

In terminal.

our project.

/Desktop/ hello-docker

mm- docker build -t hello-docker

To see all the images in our PC.

comm- docker image ls.

To run our image.

mm-docker run hello-docker

To pull / download docker image from dockerhub to our PC.

Comm- docker pull Username/ Applica^n name
image name

Kubernetes.

1. What is Kubernetes –

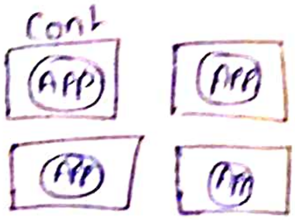
- Open source container orchestra^n tool /management

- Developed by Google.

- Helps manage containerized applications in different deployment environments

  like - physical machines.
  virtual machines.
  cloud. env.

2. Problems Kubernetes solves –



- Trend from Monolith to Microservices.
- ↑se use of containers.
- Demand for a proper way of managing those 100s of containers.

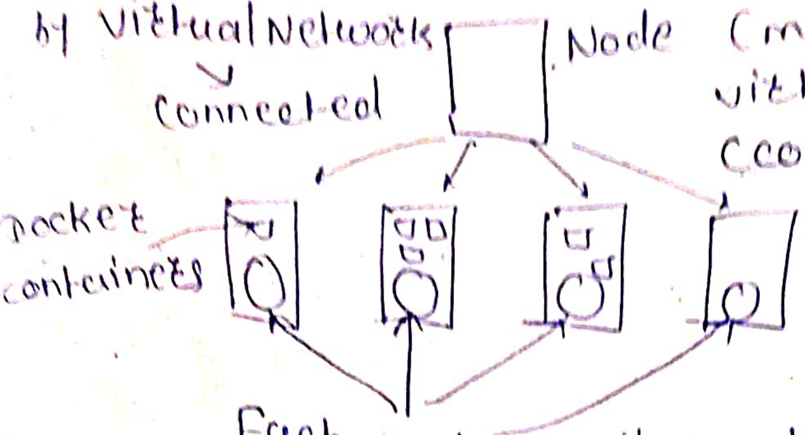
3. What features do orchestra^n tools offer?

- High Availability or no downtime.
- Scalability or high performance.
- Disaster recovery – backup & restore.

Architecture :-

Runs kubernetes Processes necessary to run & manage cluster Properly

Atleast one (master node) virtual / Phy machine (control plane)



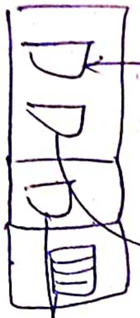
} worker nodes here your applicans are running.

Each node having a kubelet process Running on it.

1° node agent

makes in possible for the cluster to talk to e'o & Run applican process or execute task on nodes.

masternode → Using this we can access cluster



1. API server → Entry pt. to K8s cluster.

UI API CLI.

2. ~~Contain~~ controller manager.

Keeps track of whats happening in cluster.

3. Scheduler

Ensures Pods placements.

4. ~~etd~~ etcd

kubernetes backing store.

Key value storage.

Data of each node & container stored cluster.

Virtual Network → creates one unified machine.

Control Plane Nodes.
(can be multiple)

handful of master processes

much imp.

Worker Nodes.

high workload.

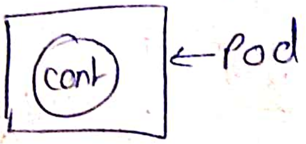
much bigger & more resources.

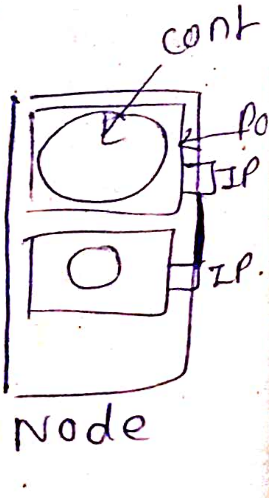
Kubernetes components.

Pod   ConfigMap   Statefulset
service   secret   Daemonset
Ingress   Deployment.   Node..

1. Pod. - Abstracⁿ of cont.

- layer / running env on top of container.
- smallest unit in kubernetes.
- Abstracⁿ over container..

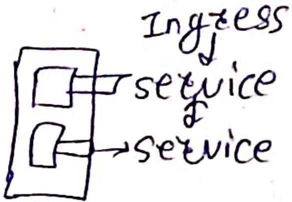




- Usually 1 Applicaⁿ cont per pod.
- Each pod gets its own IP address.

2. service — communica^{n}.

- Permanent IP address of Pod.
- lifecycle of pod & service not connected



3. Ingress. Route traffic into cluster.

- Instead of service request first goes to ingress.

4. ConfigMap — ext configura^{n}.

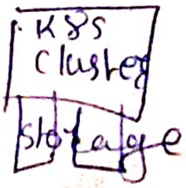
- To change the database URL.
- External configura^{n} of your applica^{n} stored.

5- secret - ext configura^{n}.

- ~~Us~~. just like config map
- But used to store secret data.
- Encrypt credentials.
- Use it as a env variables or as a properties file.

6.- ~~Volume~~. — data persistence.

- Data storage. on local machine
- or remote, outside of K8s cluster.



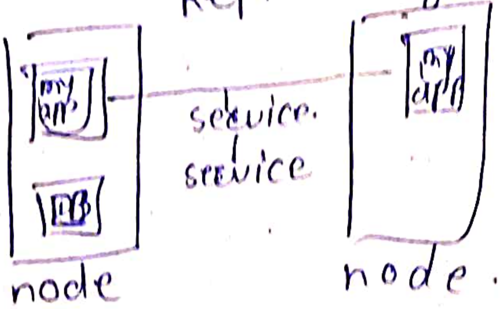
7. Distributed systems.

- If pod fails
- Then we have copy.

8. Deployment $\xrightarrow{\text{Replican}}$ for stateless Apps.

- Blueprint for "my-app" Pods.

Replicating pods.



- cannot replicate data.

9. Statefulset (sts) → for stateful Apps or DBs.

- To avoid data inconsistency → Replican
- read/ write restricns