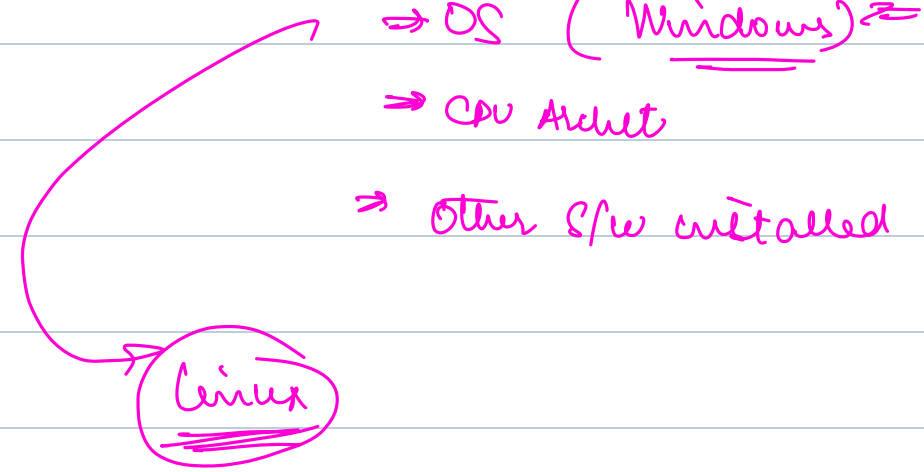
1. Revise why docker
2. Why Kubernetes
   - History of Kubernetes
   - Architecture of Kubernetes
3. Practical
   1. Product Service
   2. User Service
   - → Create docker images
   - → push the images to docker hub
   - → deploy using Kubernetes.

7:30 - 8 AM

WHY DOCKER

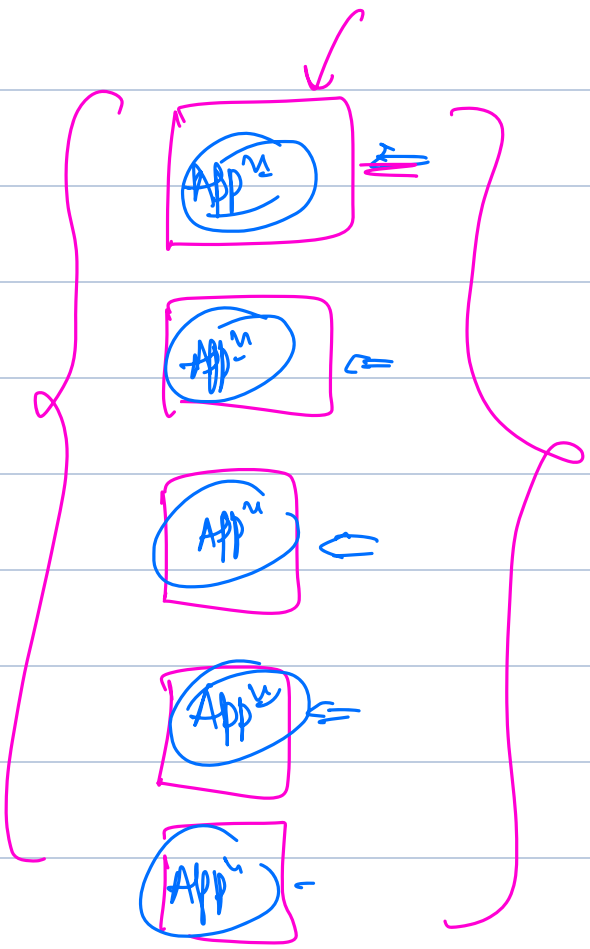
a) Diff environments



in windows: "C:\Home\ ____ "

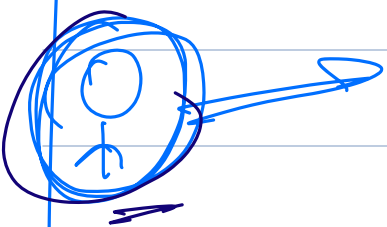
ubuntu: "/home/ var "

→ Diff to make sure that the env in which the appⁿ will finally run is the same as the env in which the appⁿ is being developed



↳ Docker: Containerization engine

If docker you didn't exist :



① Write a doc that can be used to teach any person on how to setup the app^{n} on a server.

Server

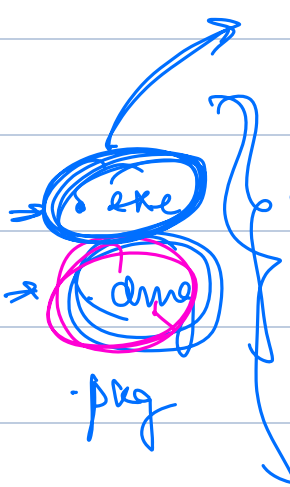
⇒ Docker is doing nothing but automating this

① Dockerfile ⇒ set of instructions to prepare your app^{n} to be deployable.

→ docker tries to keep envs same as possible

⇒ docker creates an image

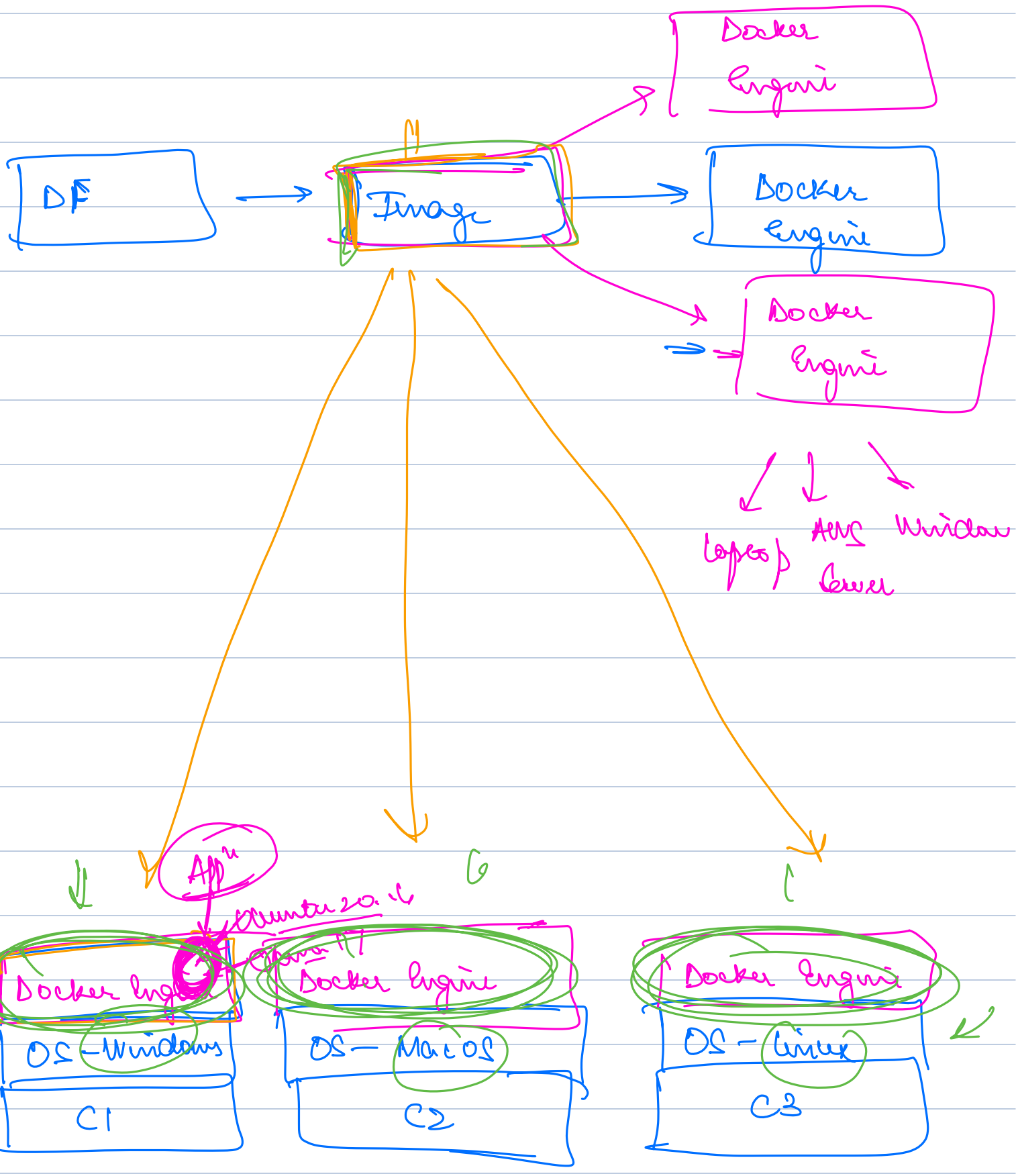
↳ contains everything that is needed to run an app^{n}



↳ Contains everything to install app^{n}

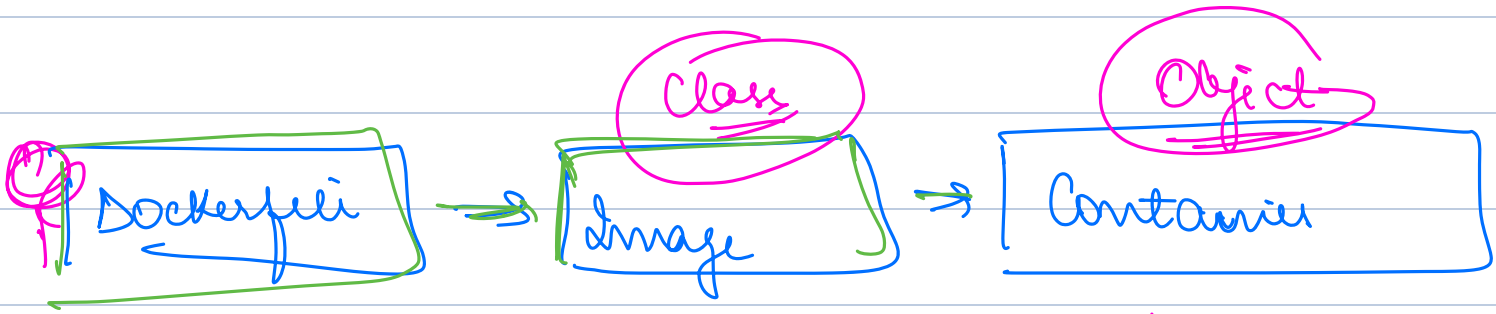
⇒ ↳ OS
↳ dependencies
↳ app^{n}

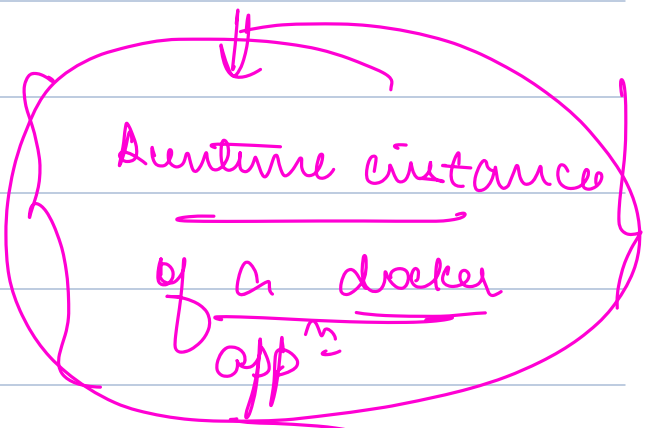
Dockerfile: Set of instructions to create docker Image ⇒ transferred / downloaded



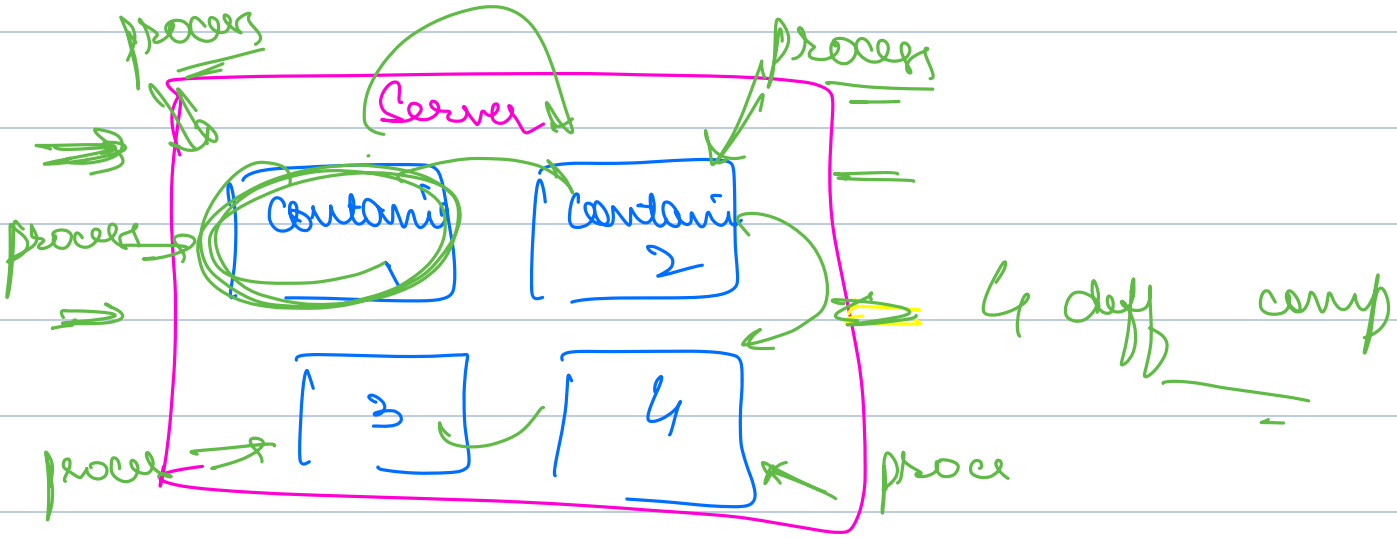
After docker engine is supplied an image, it creates a process where it runs the image

↳ isolated env with exact same config in image

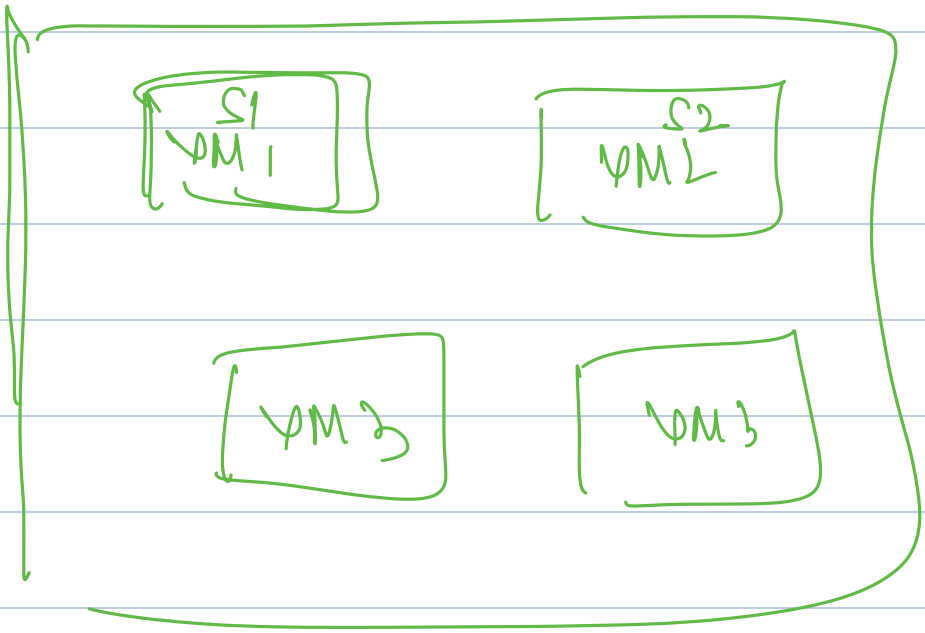




⇒ for all practical purpose it acts like a sep comp



4 diff comp

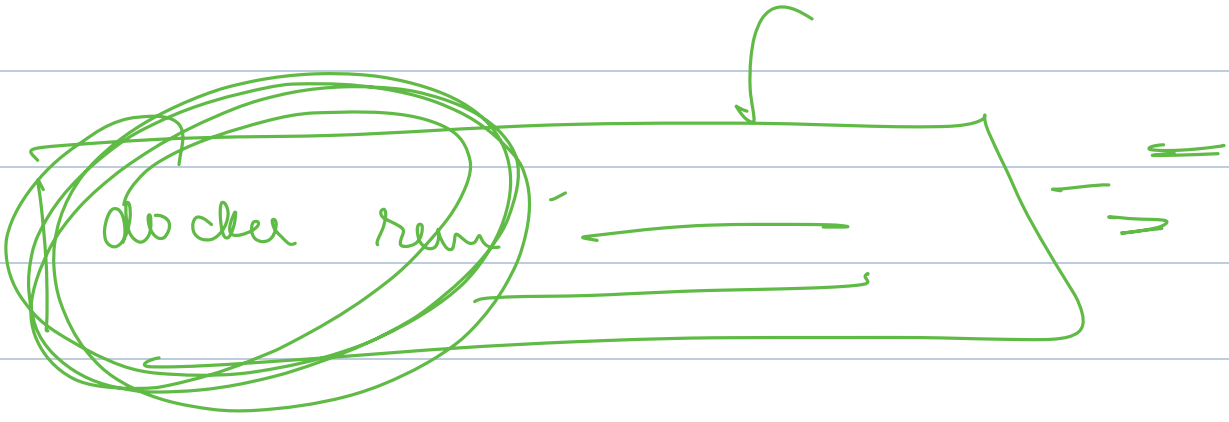


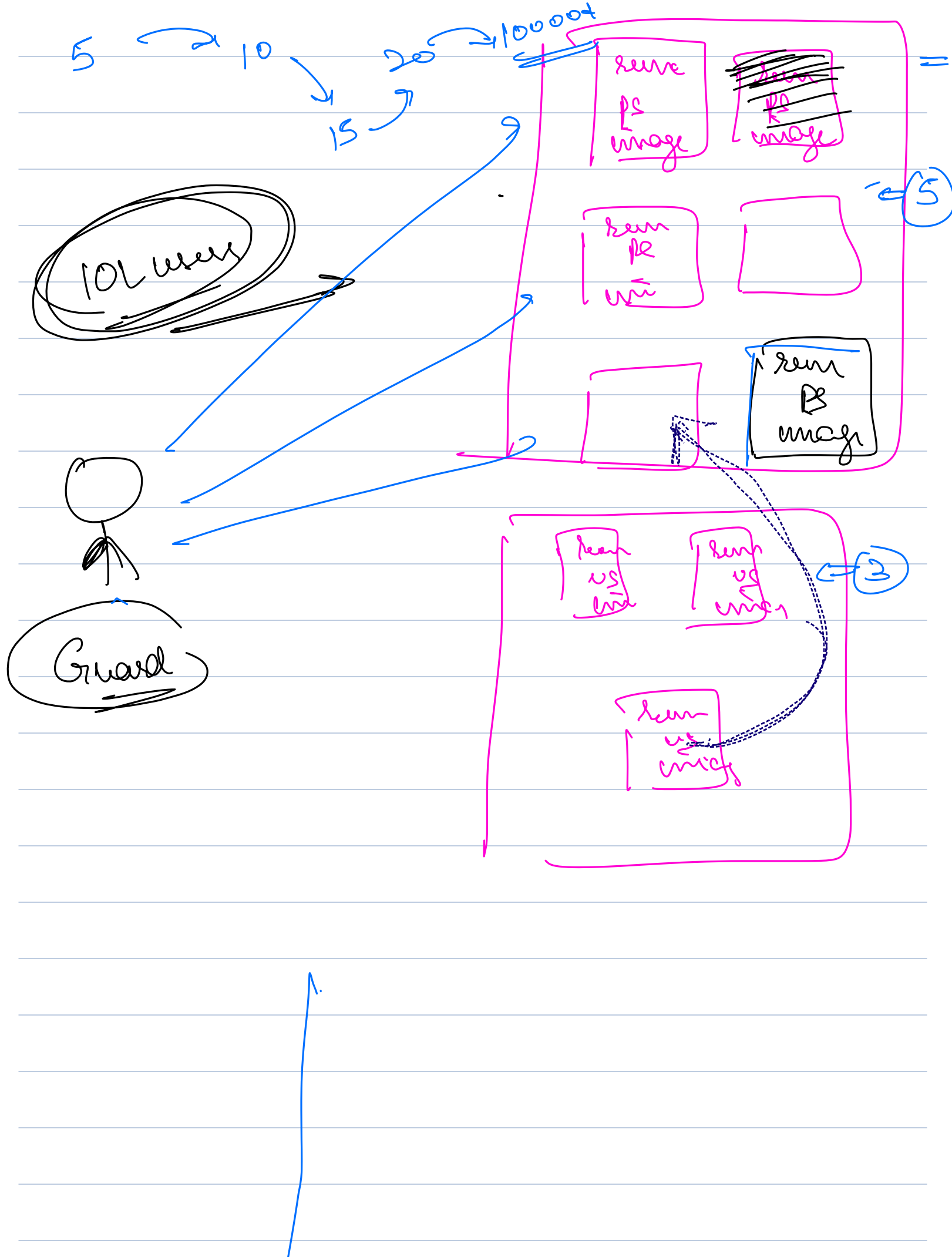
In Prod^{n}

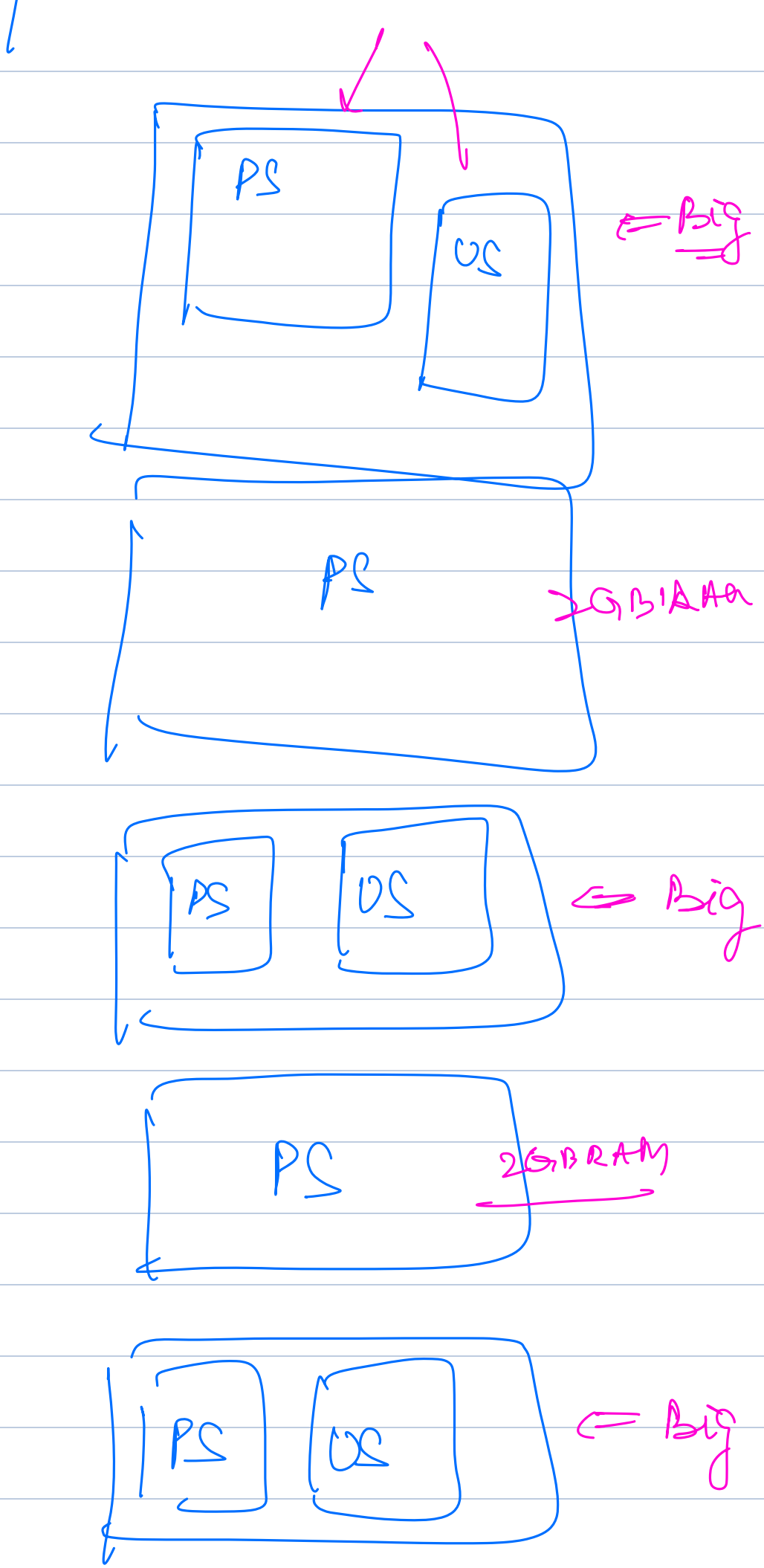
many instances of every app^{n} will be running

Product Service → 5 ~~50~~ diff servers

User Service → 3 ~~10~~ servers



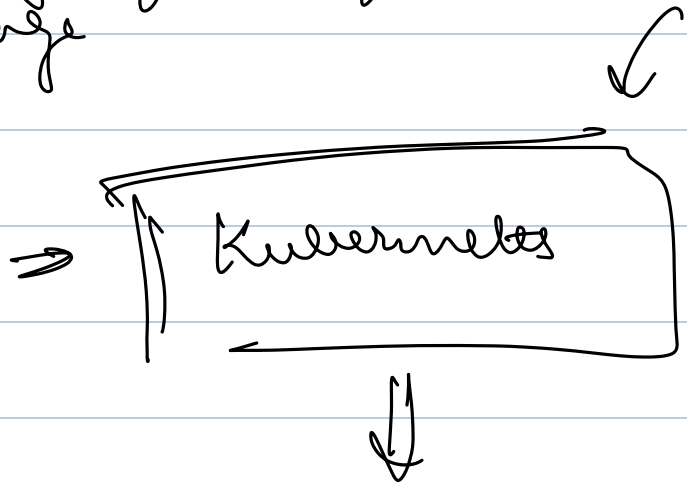

5
10
15
20
100004
10L users
Guard
5
3


PS
OS
Big
PS
2GB RAM
PS
OS
Big
PS
2GB RAM
PS
OS
Big

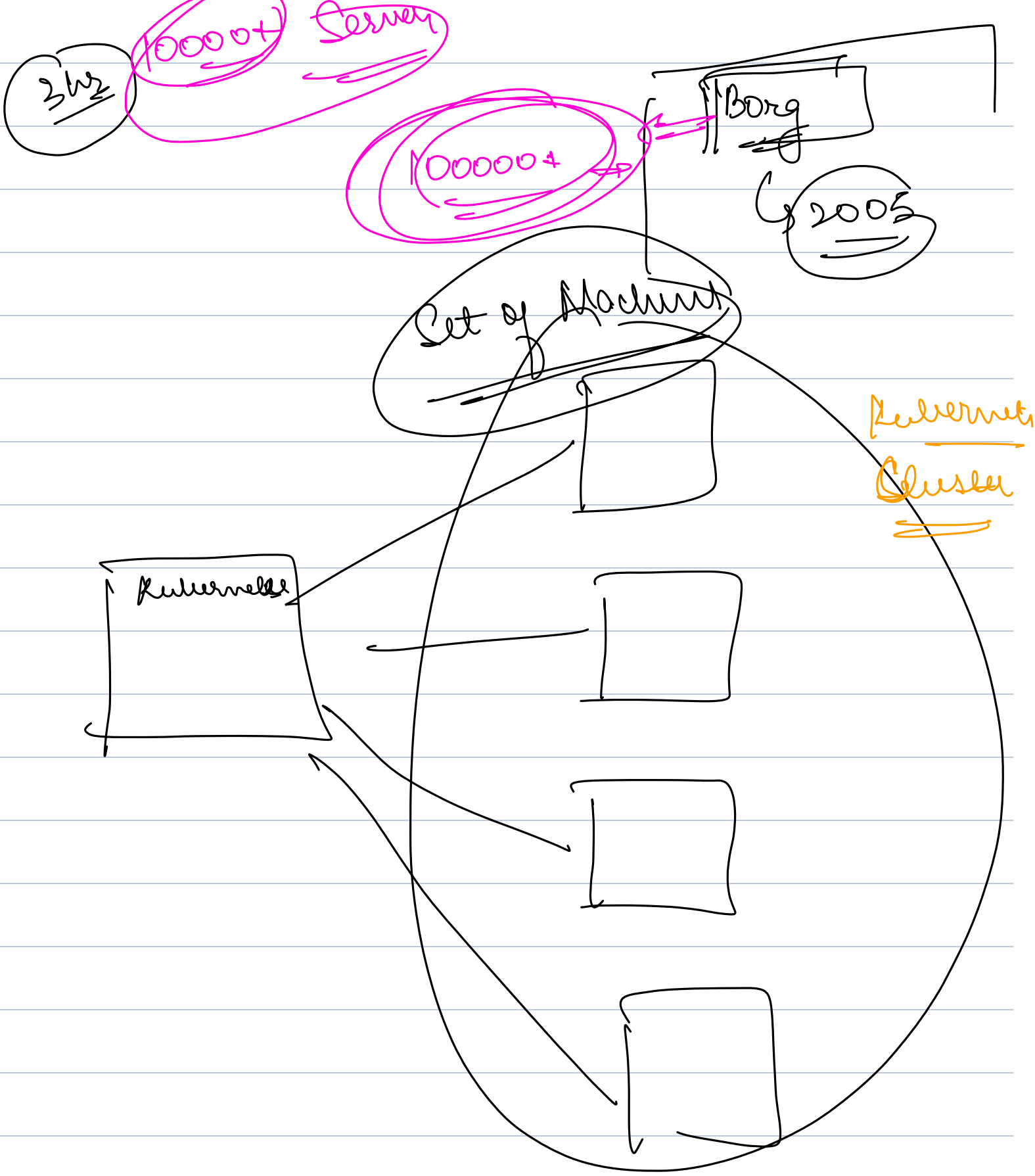
⇒ optimal use of each service

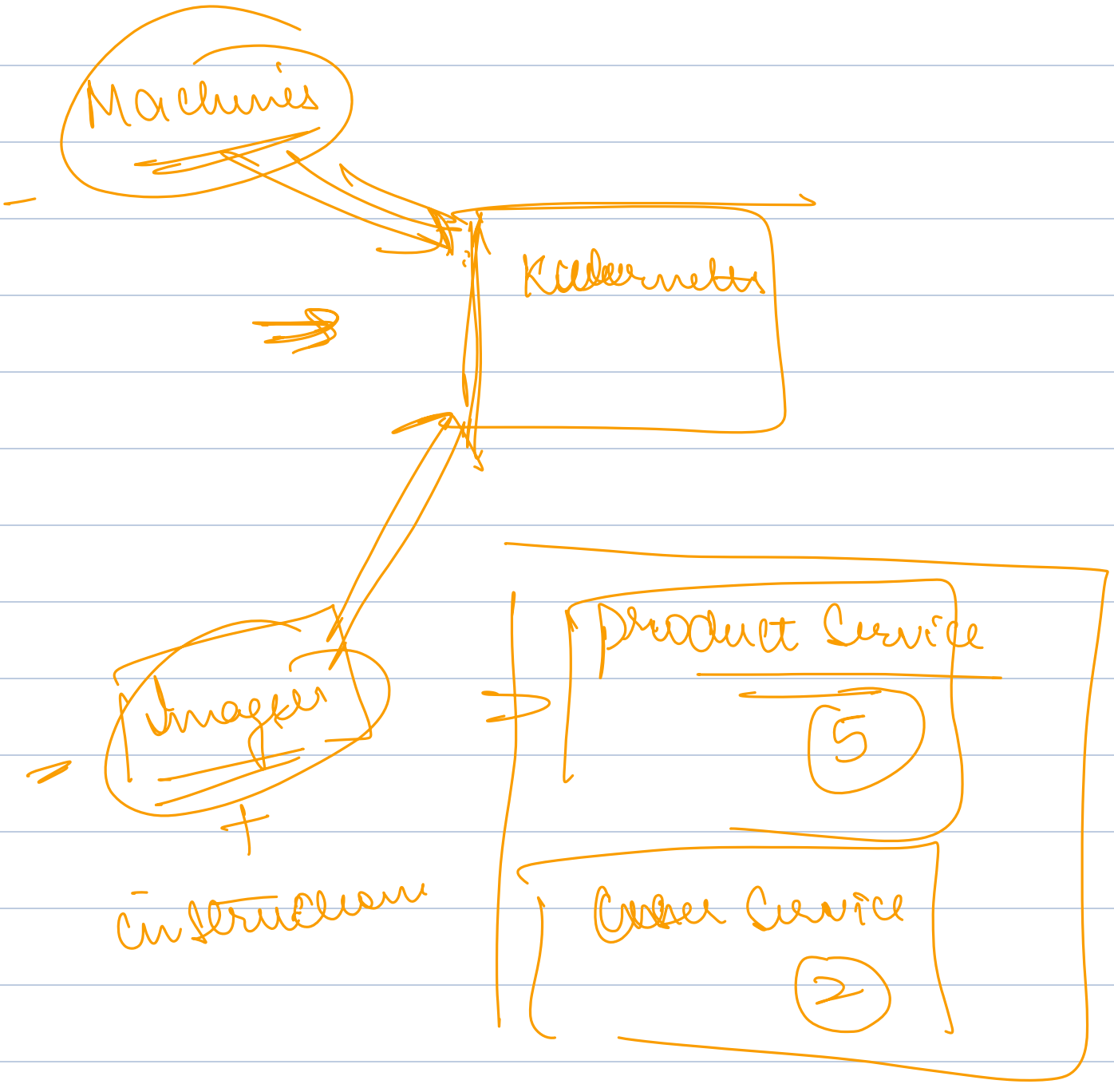
(1) Monitoring diff instance of app^n

(2) Allowing for efficient resource usage



⇒ Orchestrator ⇒ Manager controls and plans the app^n distribution across multiple service.


3 Yrs
10000+ Server
100000+
Borg
2005
Set of Machines
Kubernetes
Kubernetes
Cluster



Heartbeat + Healthcheck

Break till 7:40

1 month
6 month

2 things

Docker Certi
Kube

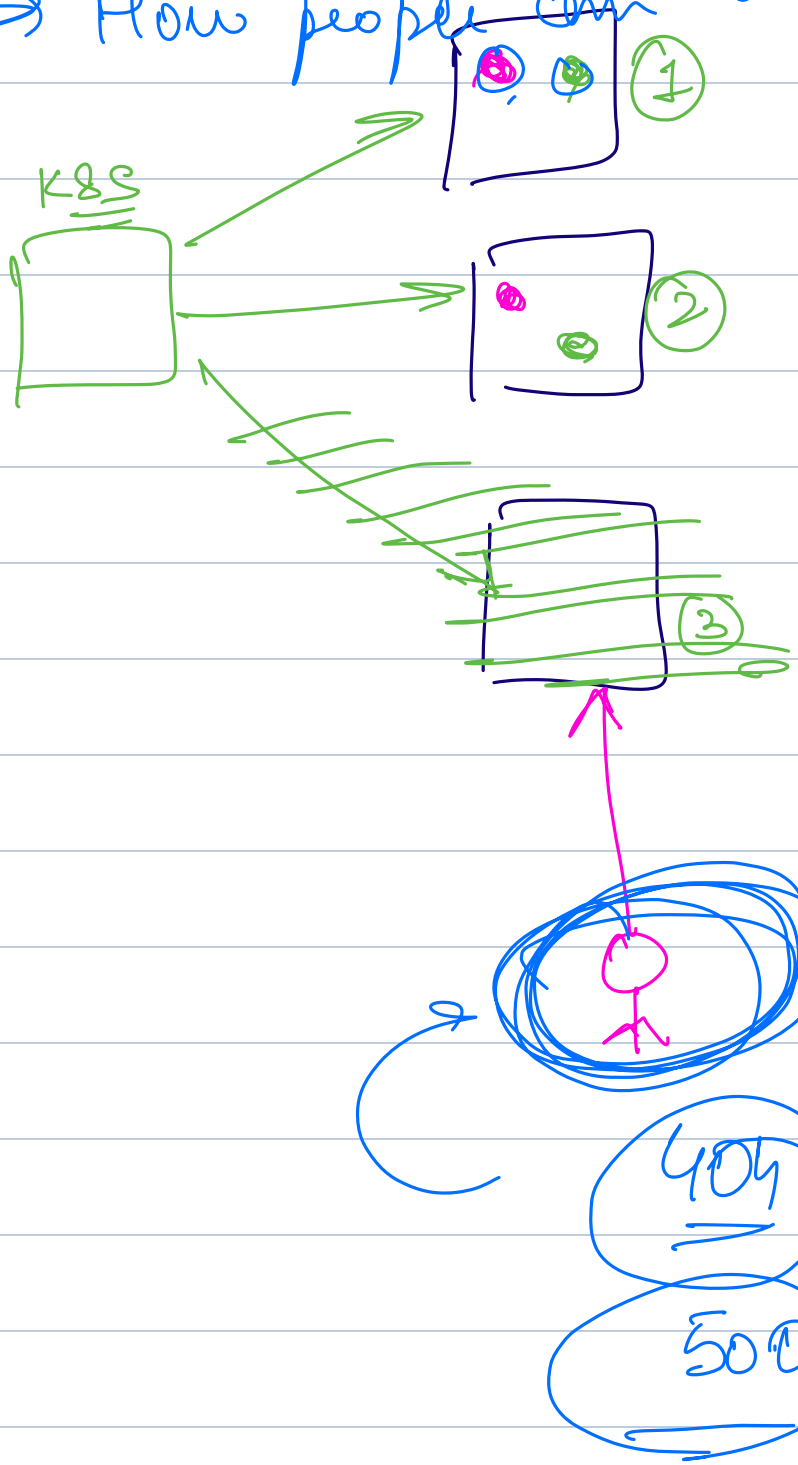
1. Deployment

config: what image
I want
how many
instances of
: what will
be the port
on which
etc

⇒ will make k8s
ensure that always
# of instances of
diff services are
running

(2) Service

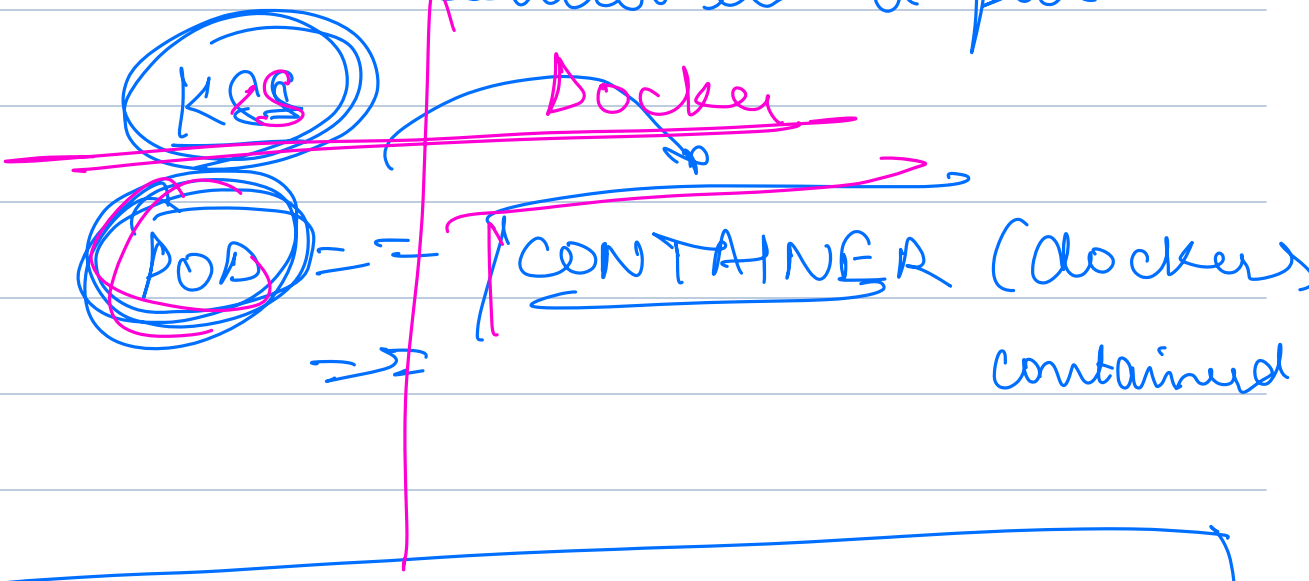
⇒ How people can interact with appⁿ in the cluster



⇒ K&S can work with any containerization engine
↳ Docker
↳ —

↳ —

→ K8s call a container a pod



→ K8s at the end is managing all pods and it can decide to move a pod anytime from one node to other

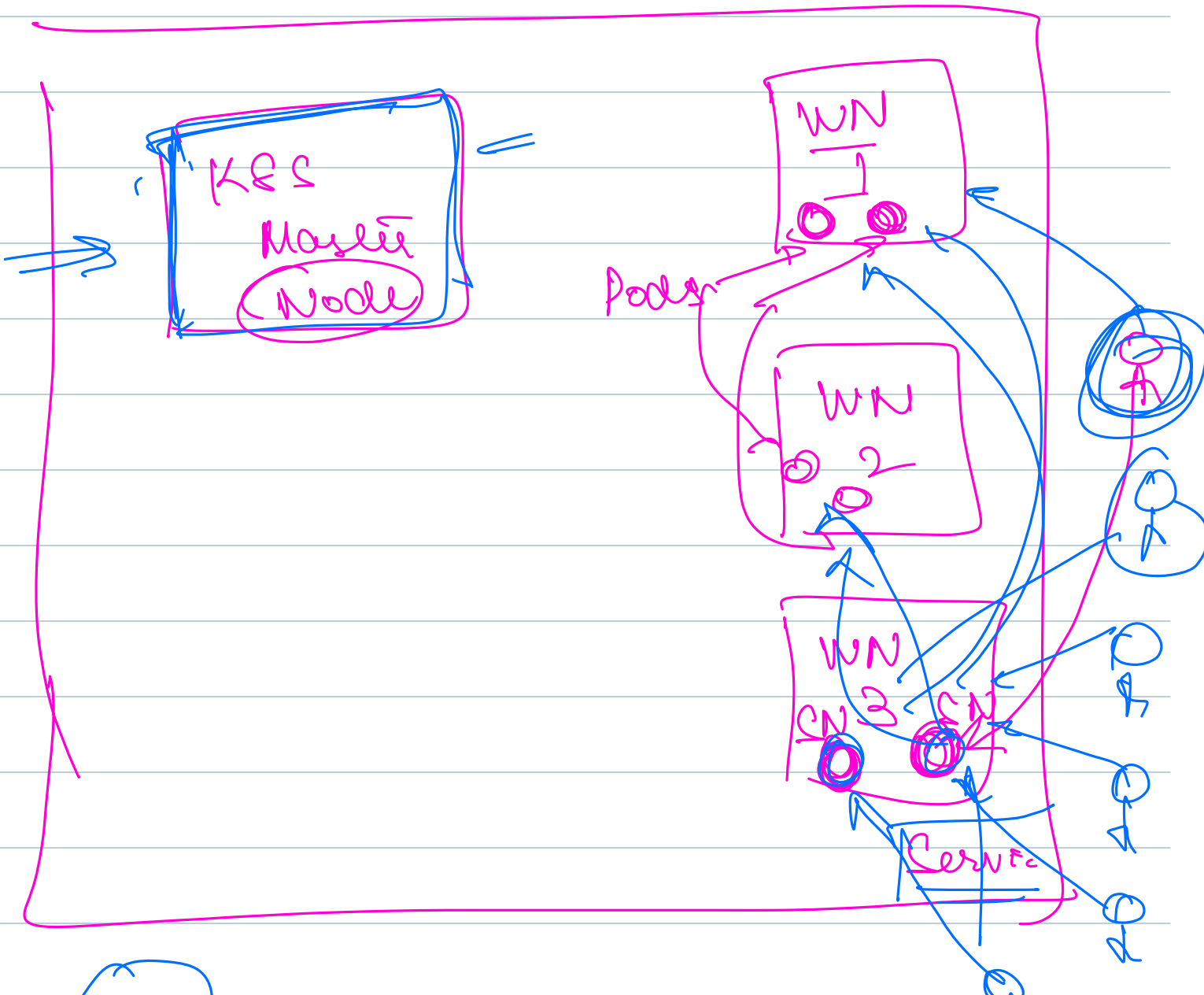
If a user is directly talking to a node ⇒ issue

Service → pod that user can talk to

⇒ Load Balancing

⇒ Auto up scaling

→ Auto down scalin



SN → Service Node

its IP is guaranteed to never change

↳ Load Balancing