Docker Swarm

starts at 9:05 pm

1. Introduction to Docker Swarm
2. Docker Swarm Architecture
3. Key Features of Docker Swarm
4. Difference between Docker Swarm and Kubernetes
5. Configuring Docker Swarm
6. Docker Swarm Backup and Restore
7. Locking Swarm Cluster
8. High Availability

Project Based Learning

Linux Ubuntu

Networking

DBMS

GIT

Docker (Networking & Storage)

Kubernetes (Container Orchestration)

CICD (Jenkins, GIThub Actions, GITLAB)

ArgoCD

Cloud - AWS

Configuration Mgt (Ansible, Chef, Puppet)

Infrastructure Mgt (Terraform)

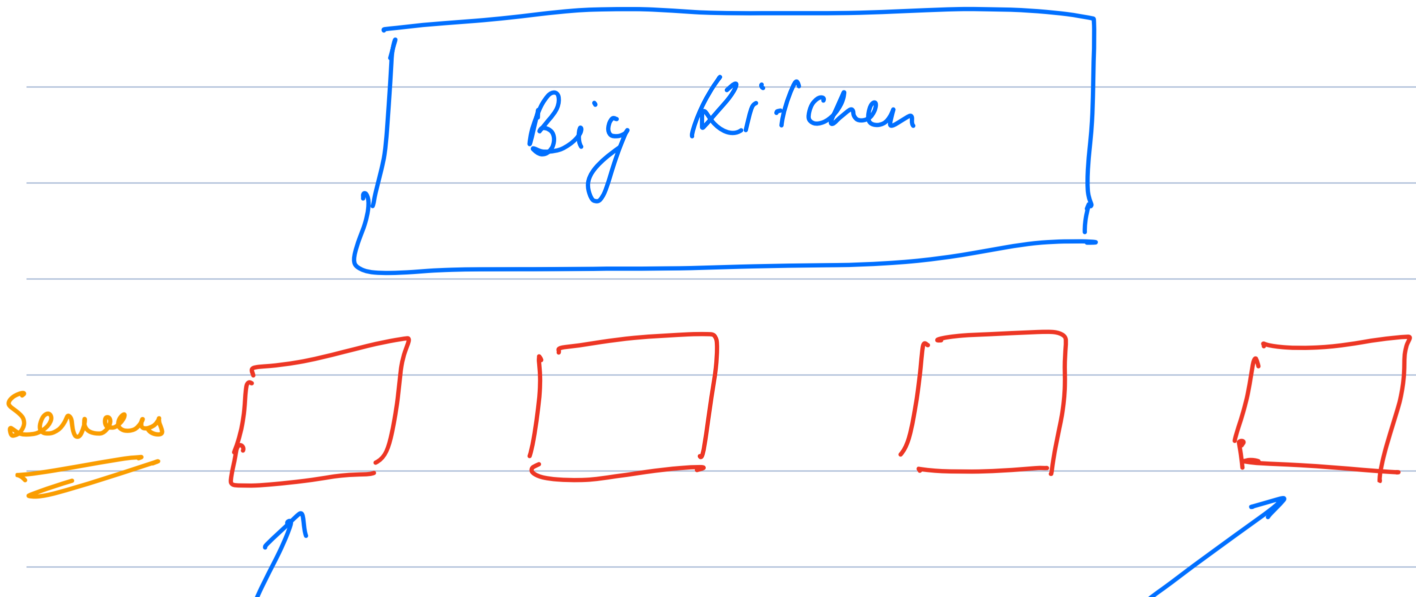
→ Introduction to Docker Swarm

Native Clustering and orchestration tool for Docker.

Container Orchestration

→ deploying
→ managing
→ scaling

} Containers



Dosa   Pasta

↘ Containers ←

→ assign tasks   (Containers

→ adjust workloads (increase or decrease)

Scaling ↙

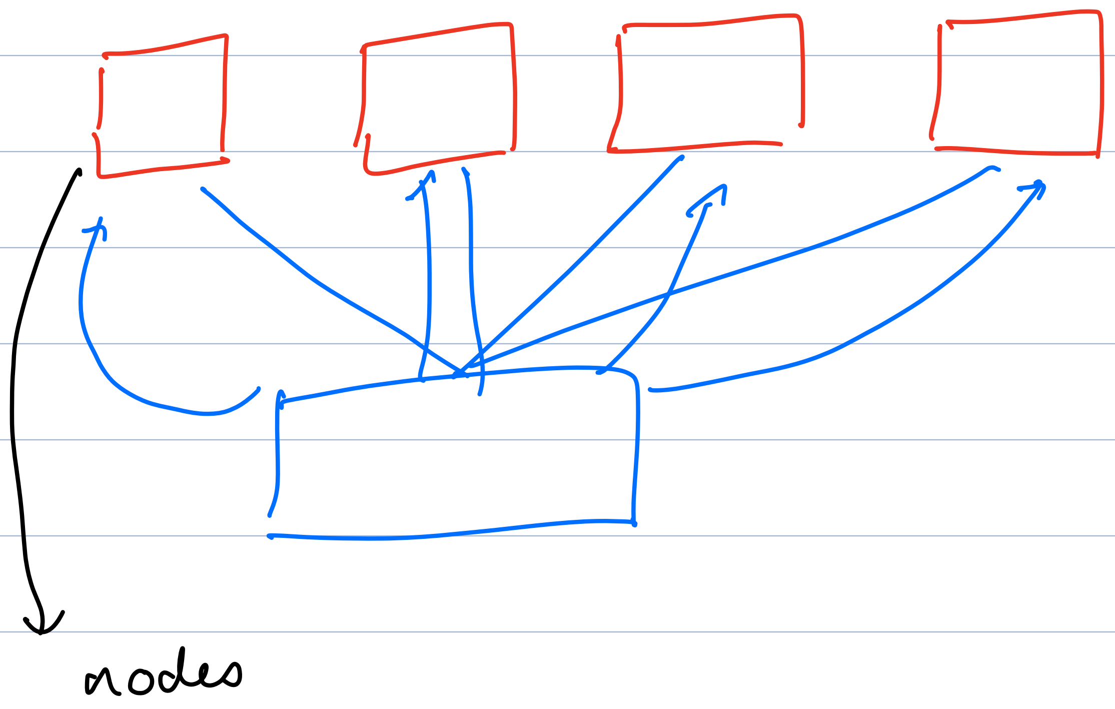
→ Replace chef or kitchen entirely)

reScheduling containers ↙

→ Communication b/w kitchens.

↙ Networking.

Orchestration is like restaurant manager which is automating all the tasks



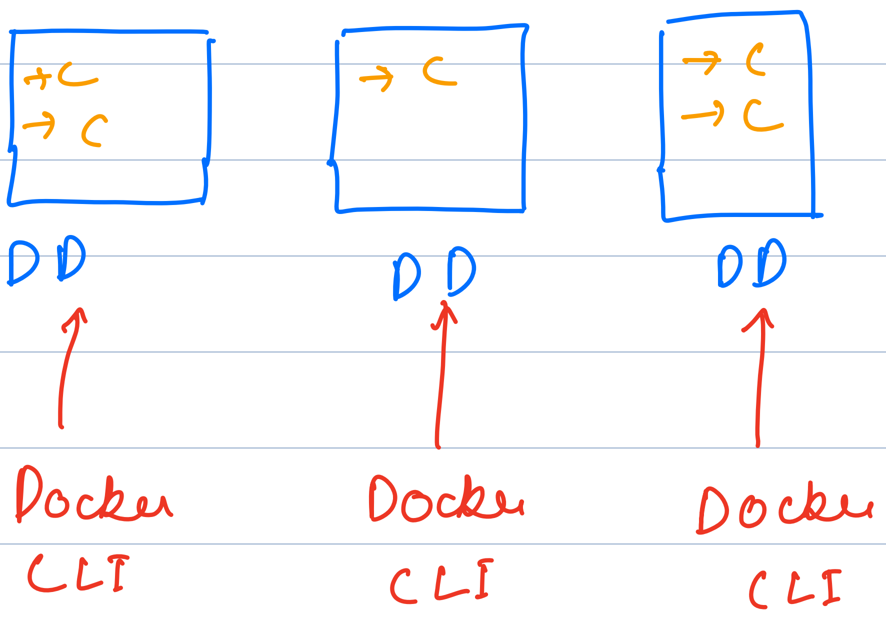
→ Services.

an abstraction in Docker that represents a single Application.

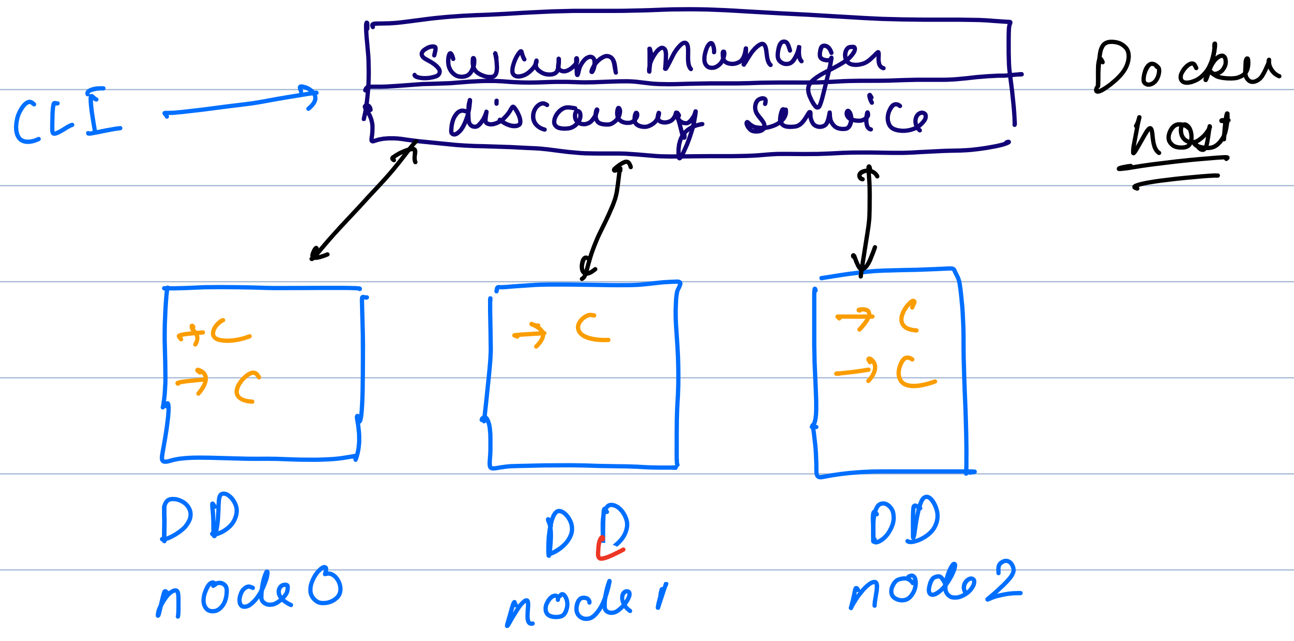
- → image
- → number of replicas (containers)
- → ports
- → volumes.

→ 5 nginx containers.

→ Architecture with Docker



→ with Docker Swarm



2 types of nodes.

1. Manager Node
2. Worker Node

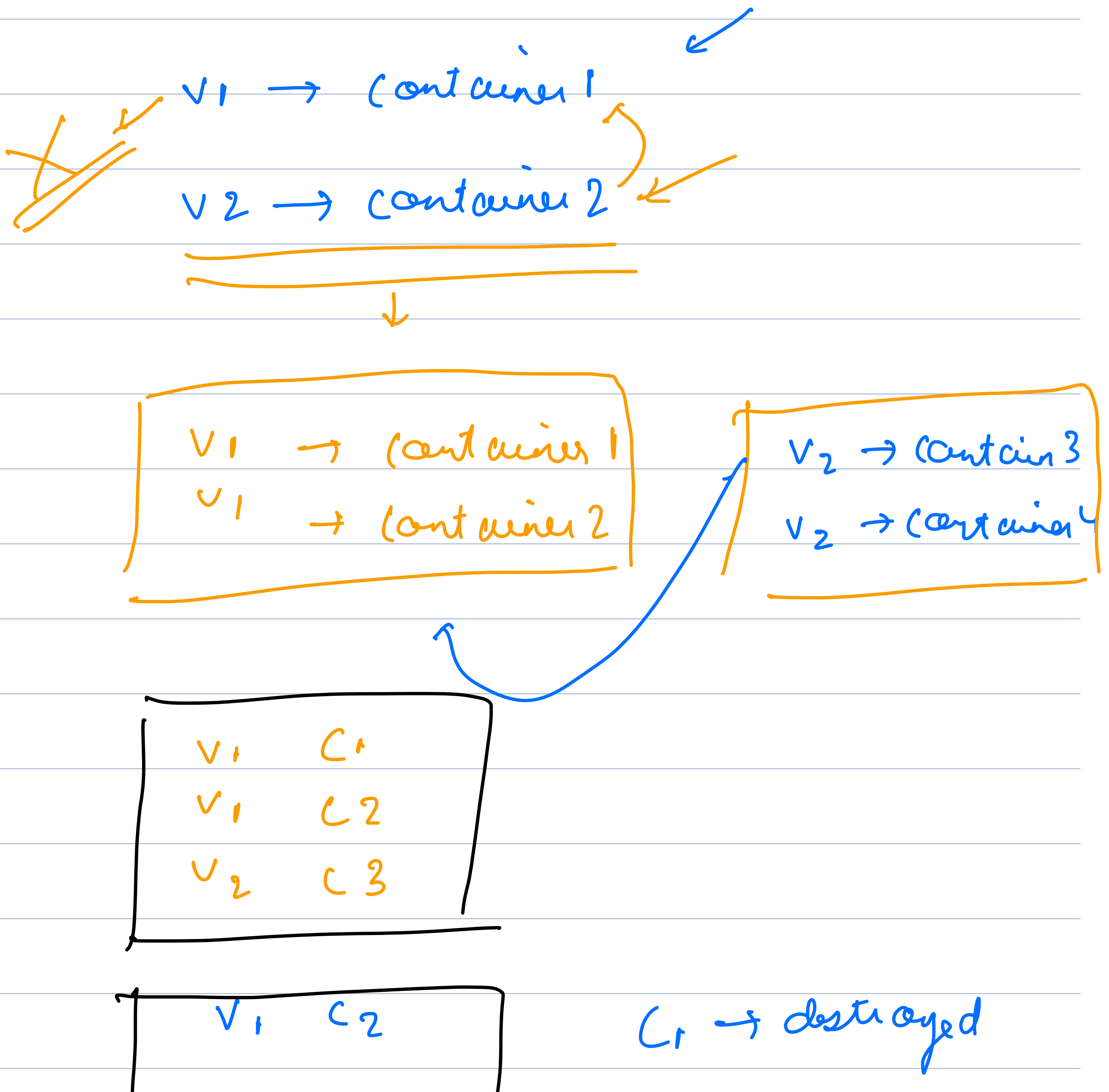
→ Swarm Manager responsibilities

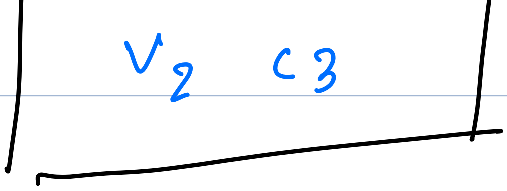
1. Scheduling services.
2. Monitoring health
3. Scaling
4. Service Discovery.

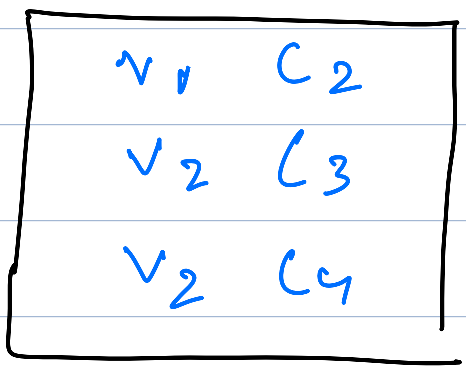
Discovery Service.

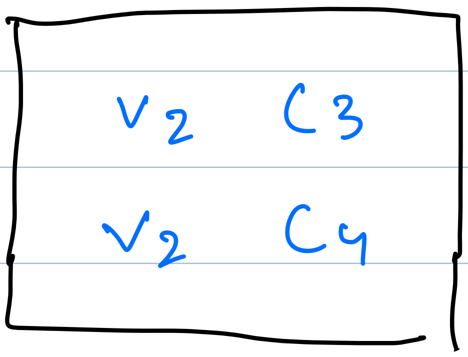
→ Key Features

1. Clustering.
2. Service Definition.
3. Load Balancing.
4. Rolling updates
5. High Availability.



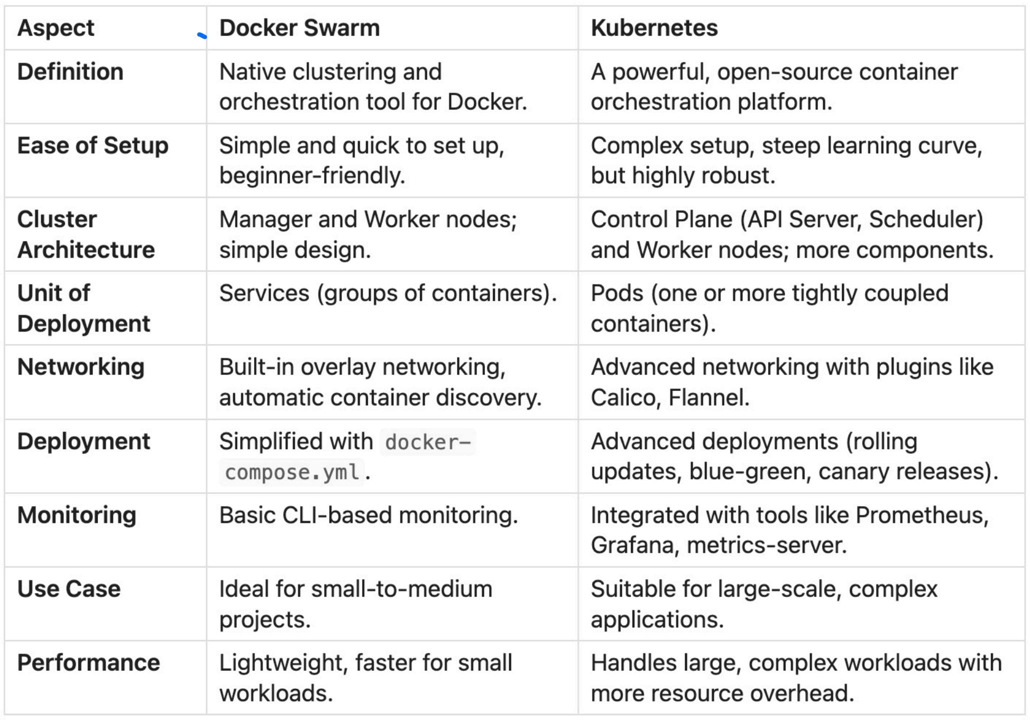






→ C_2 destroyed.

→ Docker Swarm vs K8s



Aspect	Docker Swarm	Kubernetes
Definition	Native clustering and orchestration tool for Docker.	A powerful, open-source container orchestration platform.
Ease of Setup	Simple and quick to set up, beginner-friendly.	Complex setup, steep learning curve, but highly robust.
Cluster Architecture	Manager and Worker nodes; simple design.	Control Plane (API Server, Scheduler) and Worker nodes; more components.
Unit of Deployment	Services (groups of containers).	Pods (one or more tightly coupled containers).
Networking	Built-in overlay networking, automatic container discovery.	Advanced networking with plugins like Calico, Flannel.
Deployment	Simplified with `docker-compose.yml`.	Advanced deployments (rolling updates, blue-green, canary releases).
Monitoring	Basic CLI-based monitoring.	Integrated with tools like Prometheus, Grafana, metrics-server.
Use Case	Ideal for small-to-medium projects.	Suitable for large-scale, complex applications.
Performance	Lightweight, faster for small workloads.	Handles large, complex workloads with more resource overhead.

→ 10:25pm

Configuring Docker Swarm

1 manager node
2 workers

```
sudo usermod -aG docker $(whoami)
```

Master Node → docker swarm init

↓

→ join token

docker node ls

Worker 1
↓
join command

Worker 2
↓
join command.

```
docker node update --availability drain qb0j8j8v9qbb1trci0shxb715
```

```
docker service create --name my-service --replicas 3 nginx
```

Promoting a worker node

```
docker node promote <node_id>
```

```
docker node demote <node_id>
```
→ Demotion

```
docker node rm <node_id>
```
→ Removing a Node from swarm

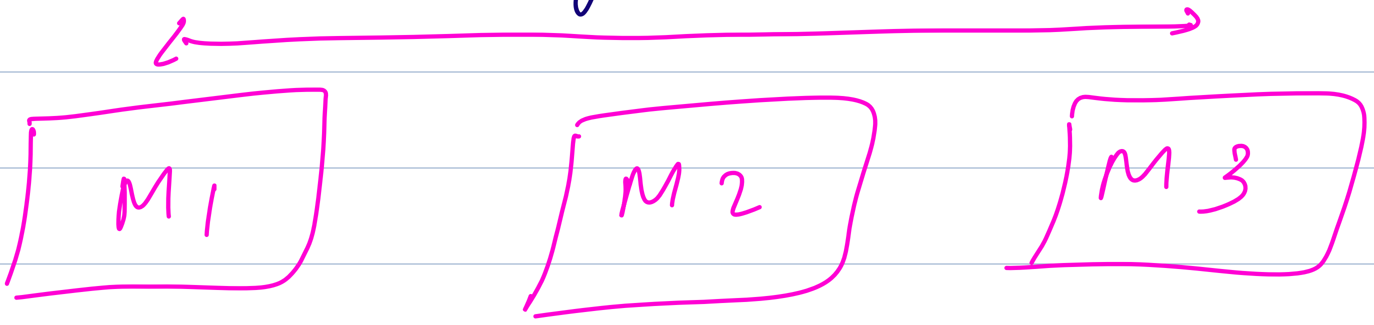
- → drain → prevents and moves tasks.
- → pause → doesn't accept new tasks
- → Active. → Allow MN to accept tasks

but keeps the current tasks running.

Docker Swarm Backup & Restore.

Raft DB → used by DockerSwarm.

→ Service definitions
→ network config
→ Node membership info
→ task assignments.



→ stopped docker.
→ Backup of raft Folder

```
sudo cp -r /var/lib/docker/swarm/raft /tmp/
```

→ started docker
→ docker service ls
// my-service

→ docker service rm my-service.

→ restoring

✓✓ my-service

M1 → looking for nodes to schedule <u>3C</u>

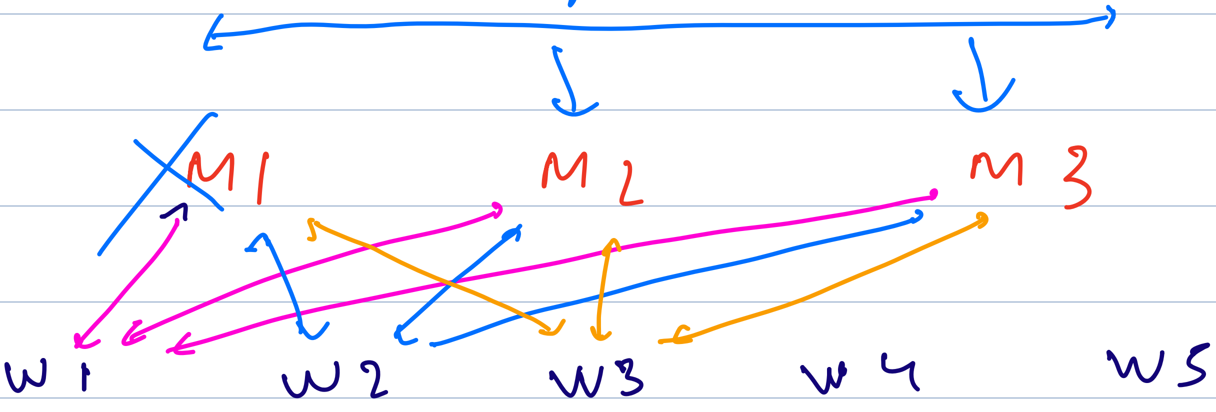
W1
→ left
→ joined

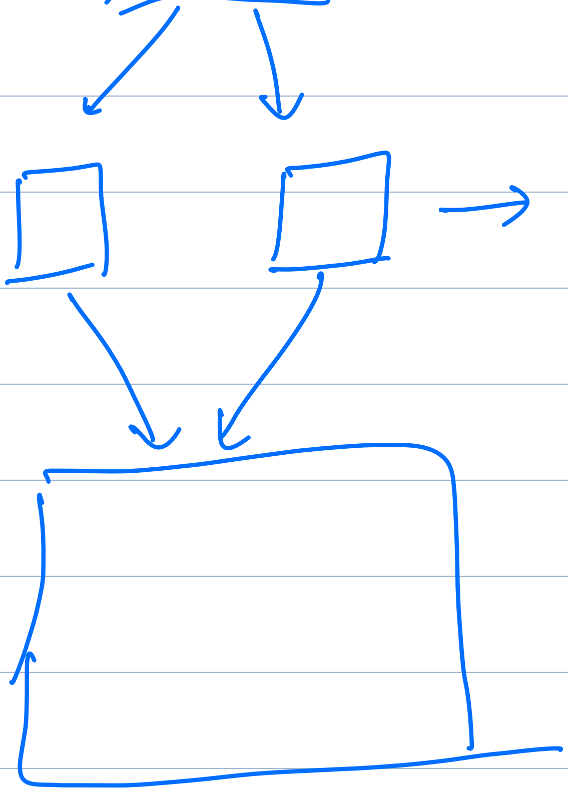
W2
→ left
→ joined

3 C1
C2
C3

→ High Availability

→ Raft





```
docker service update --image new_image_version my_service
```