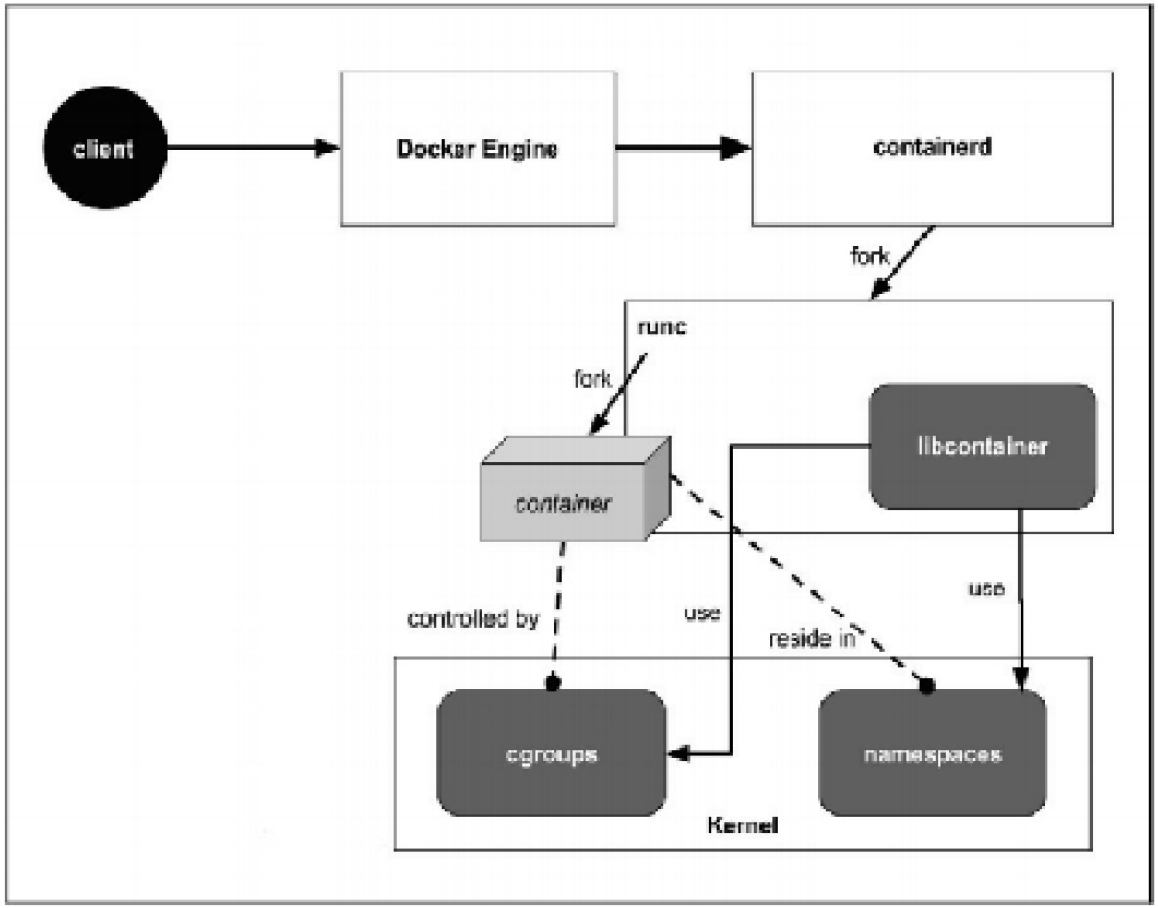
# Sisteme distribuite

Mihai Zaharia

Cursul 8

# runC

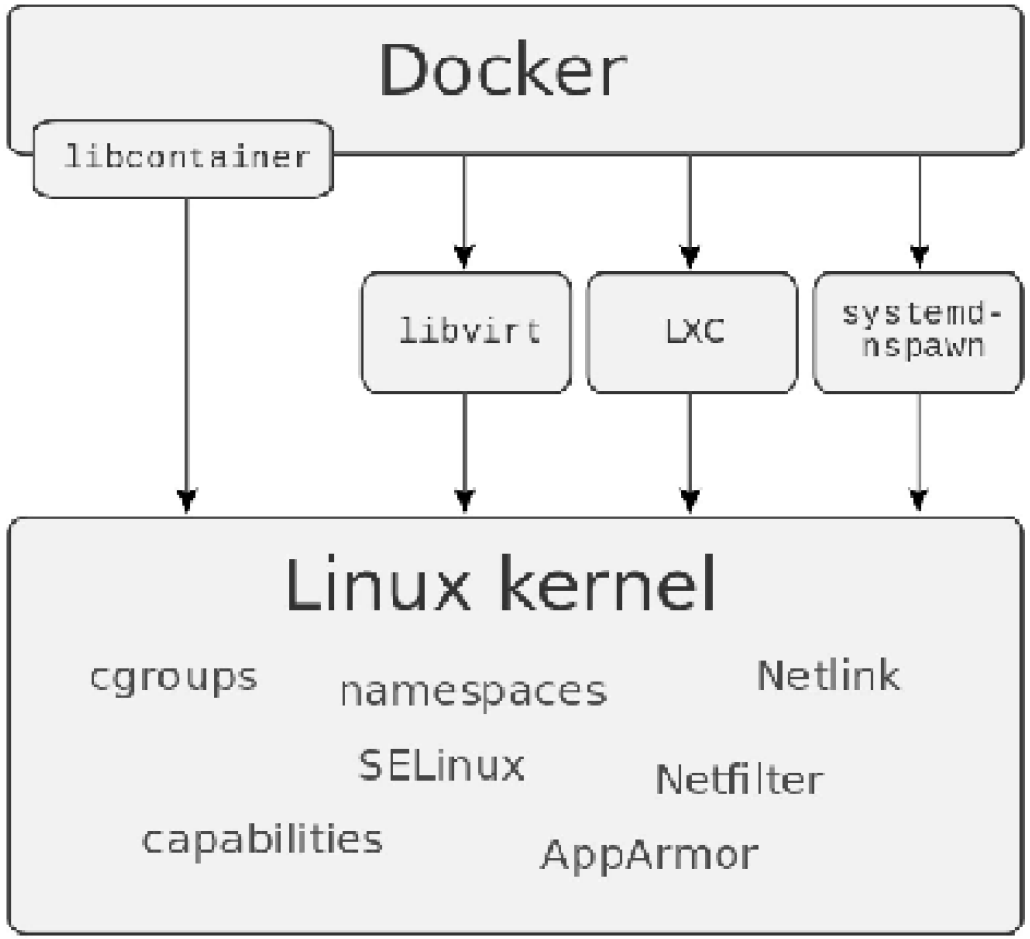



Ce oferă Linux pentru containerele docker

# Detalii rețalie mașină Docker - Linux

## Spații de lucru utilizate de mașina Docker

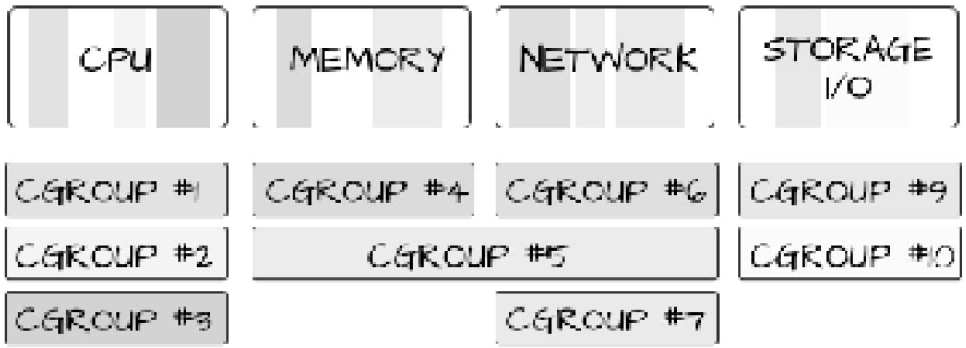
- PID
- NET
- IPC
- MNT
- UTS



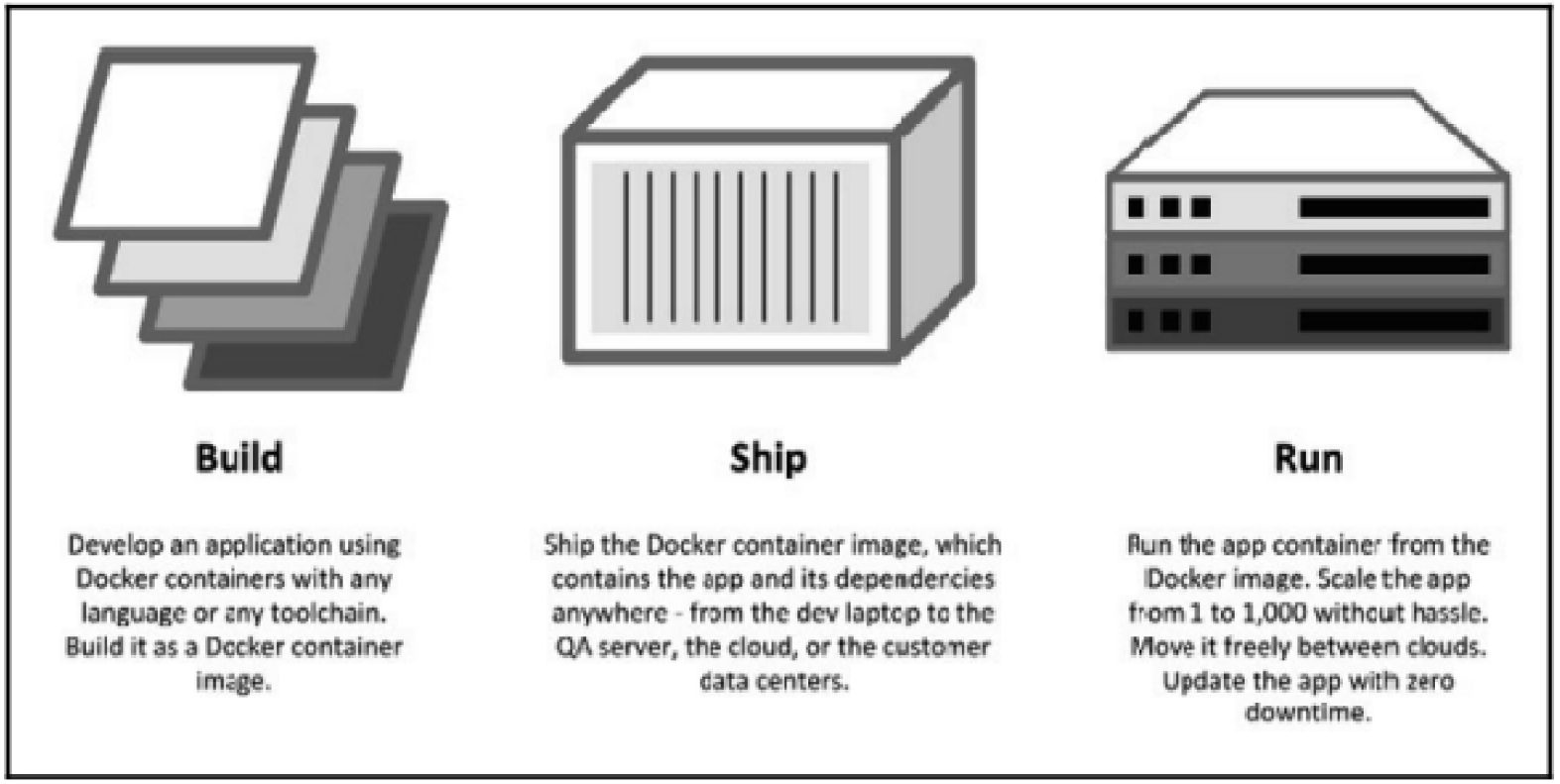
# Izolarea resurselor - detalii

**Cgroups - Izolare și enumerare resurse sistem**

- cpu
- memory
- block i/o
- devices
- network
- numa
- freezer

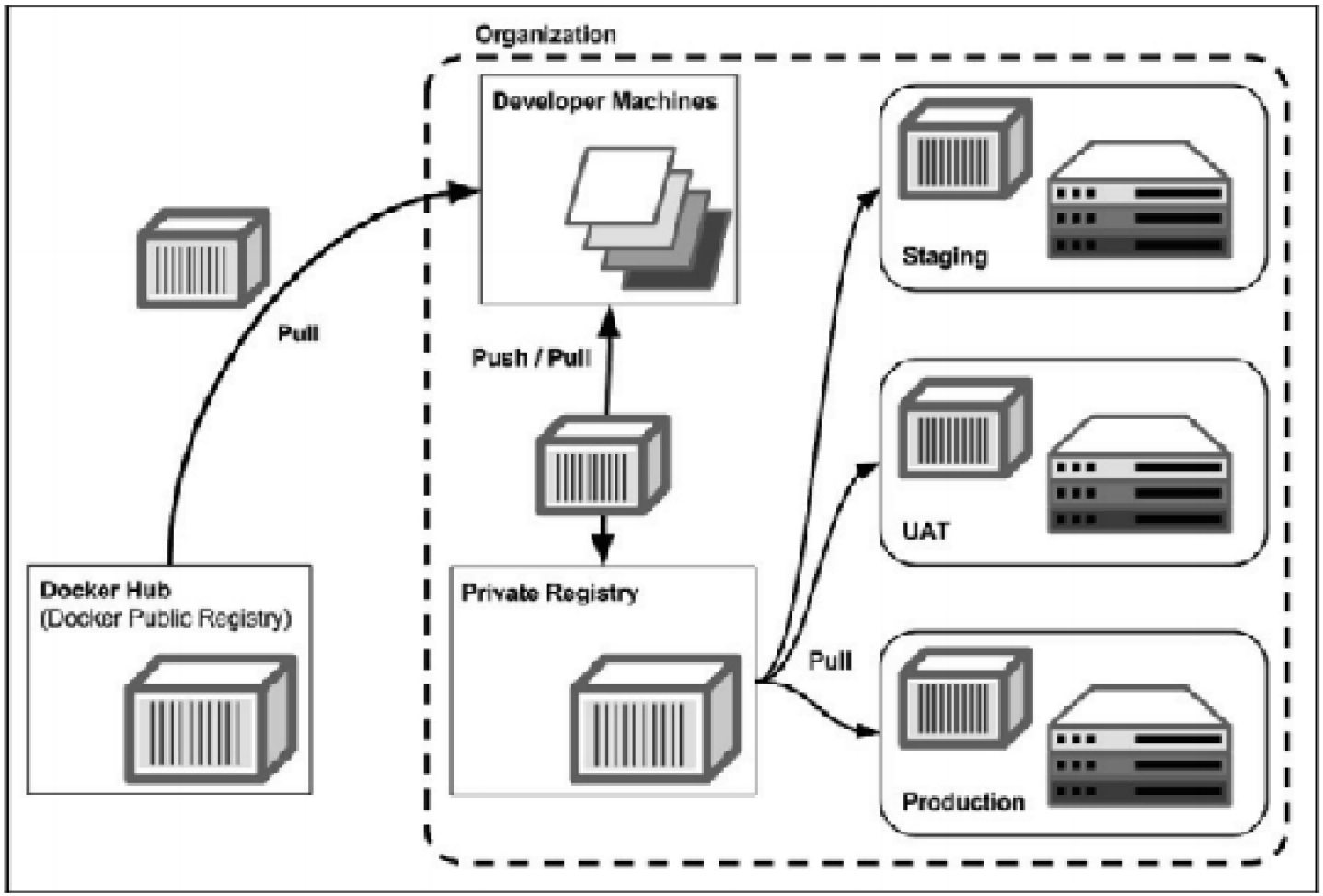


# Filozofia Docker



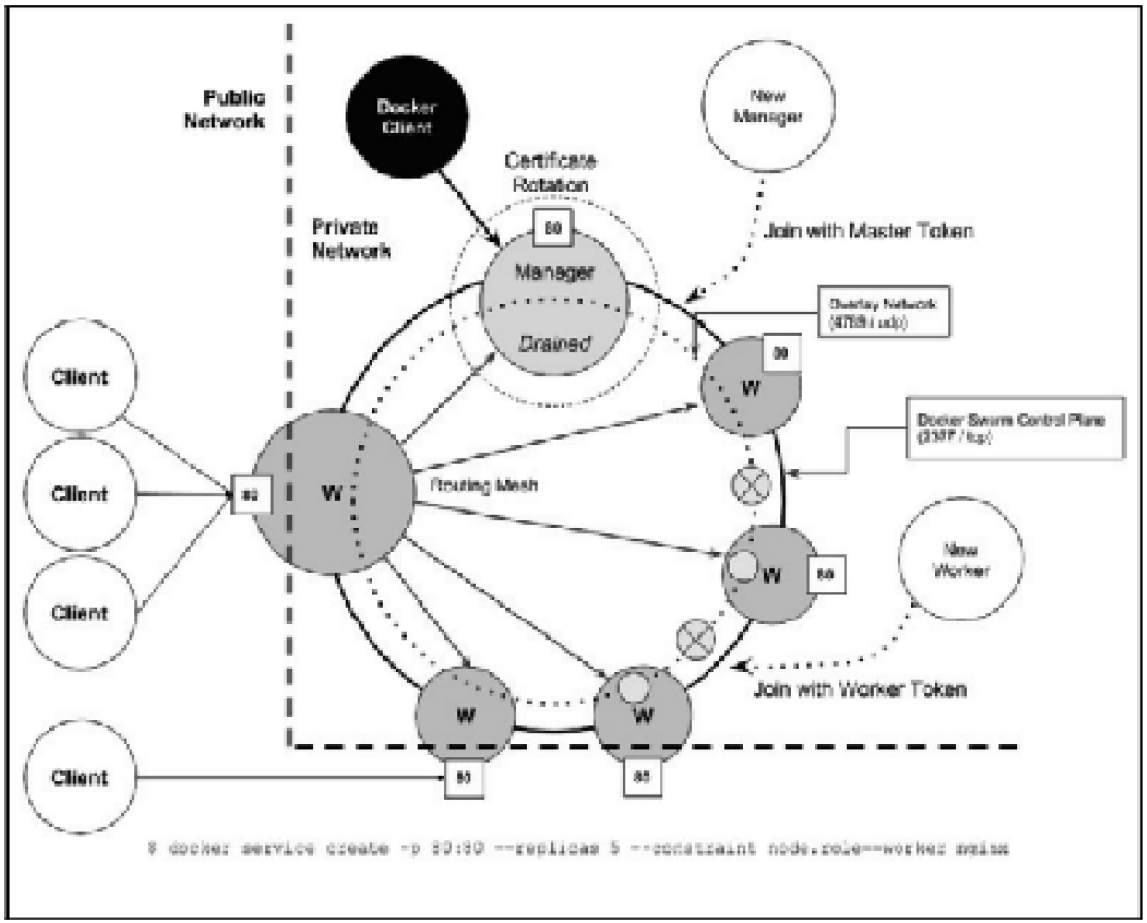

Build-Ship-Run

# Gestiunea imaginilor Docker




procesul de push/pull

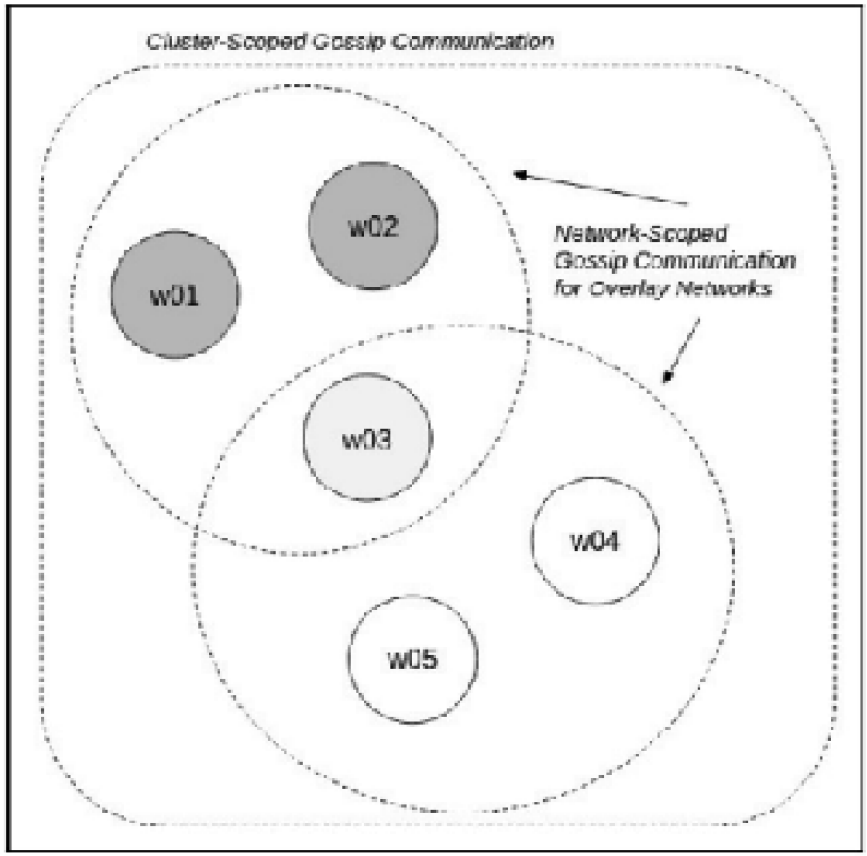
# Servicii și sarcini




- --replicas
- --name
- *ingress*
- docker node update --availability drain mg0

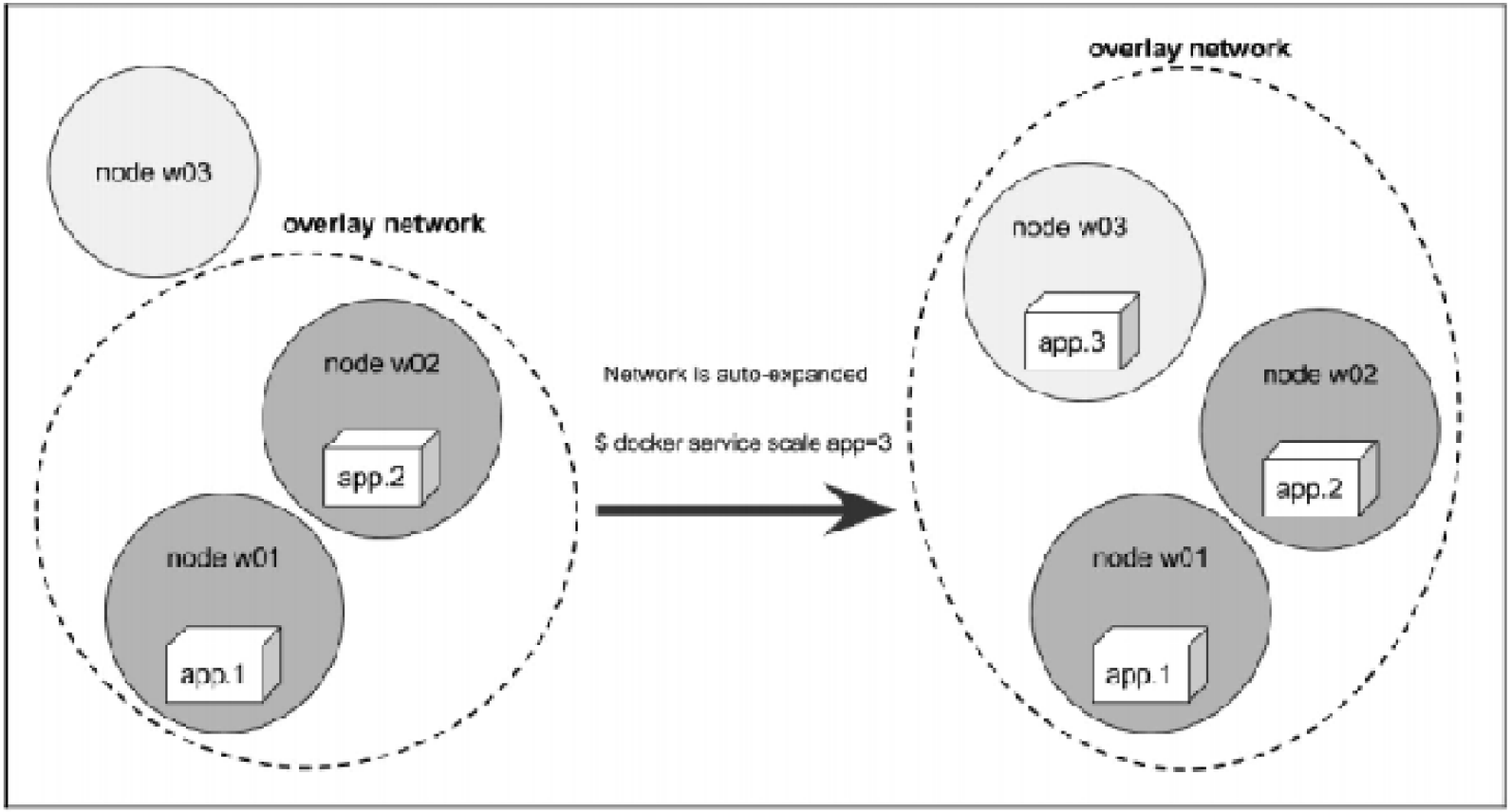
# Bârfa în Docker (nu numai la Români)

- docker inspect web




mecanismul de bârfă (gossip) utilizat
pentru comunicarea inter-rețea în roi (swarm)

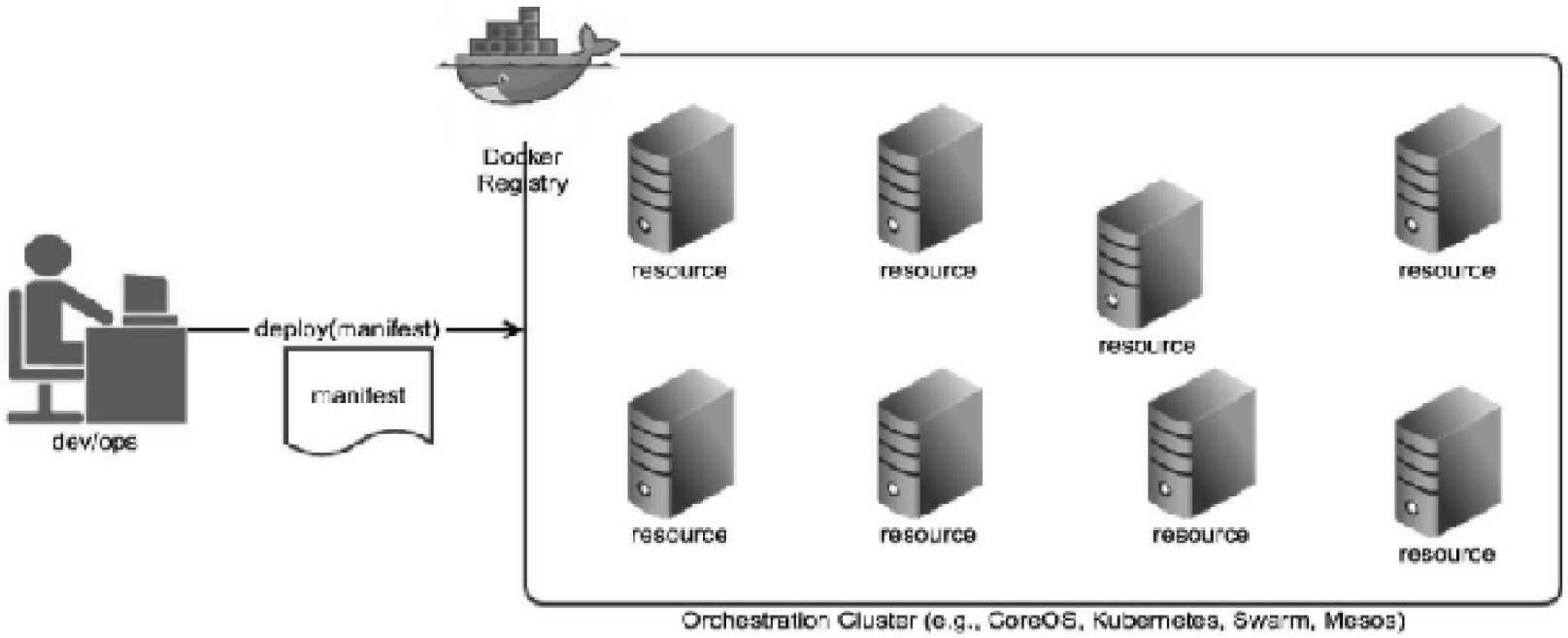
# Multiplicarea/scalarea unui serviciu




replicare în swarm

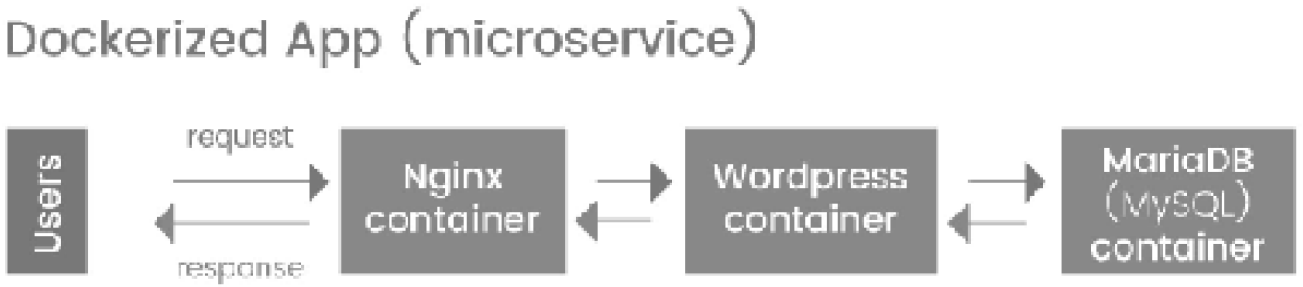
docker service scale

# Lansarea unui container Docker




pe baza instrucțiunilor din manifest

# Cum se pot containeriza microservicii



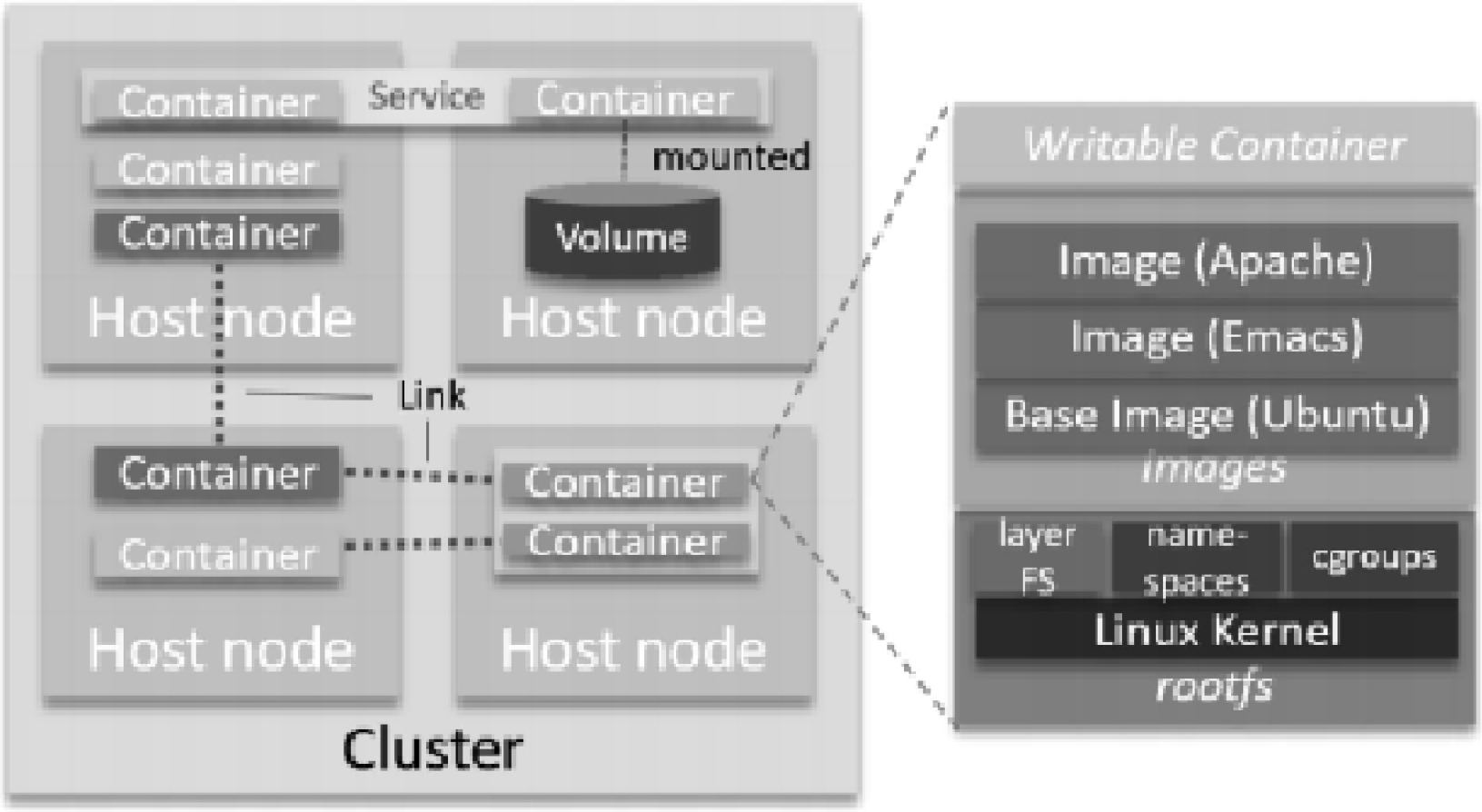
# docker-compose.yml

- version: '2'
- services:
- nginx:
- build: nginx
- restart: always
- ports:
- - 8080:80
- volumes_from:
- - wordpress
- wordpress:
- image: wordpress:php7.1-fpm-alpine
- environment:
- WORDPRESS_DB_HOST: mysql
- WORDPRESS_DB_PASSWORD: example
- mysql:
- image: mariadb
- environment:
- MYSQL_ROOT_PASSWORD: example
- volumes:
- - ./demo-db:/var/lib/mysql

# deci în esență cum fac o aplicație

1. creez un microserviciu intr-un limbaj/tehnologe
2. il testez
3. creez o imagine de container
4. creez o compozitie de servicii conform proiectarii anterioare (model arhitectural + combinatii de modele de proiectare specifice

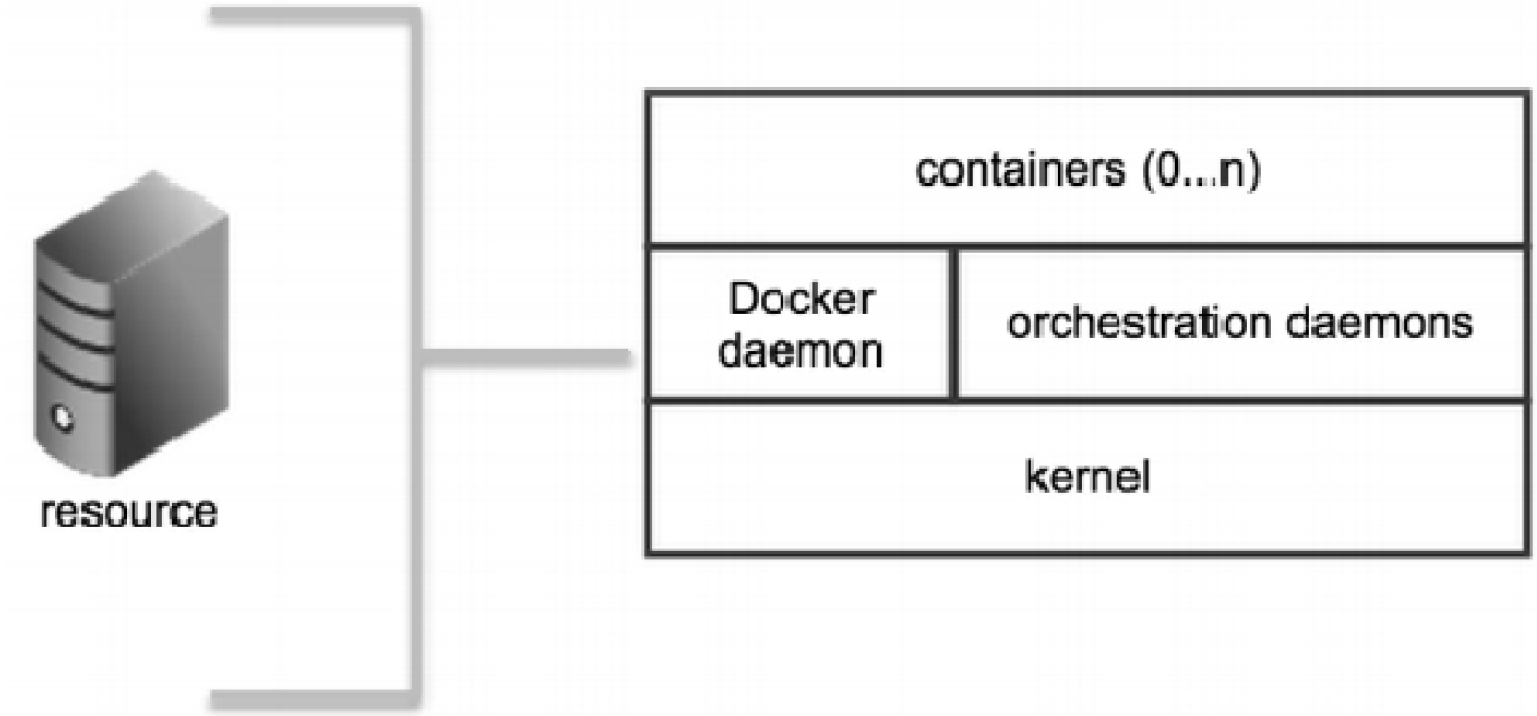
# Cluster de containere



# Containere în nor

- gestiune dependențe
- agenți pe ciclul de afaceri
- PaaS & containere

# Cum intervine orchestrarea?

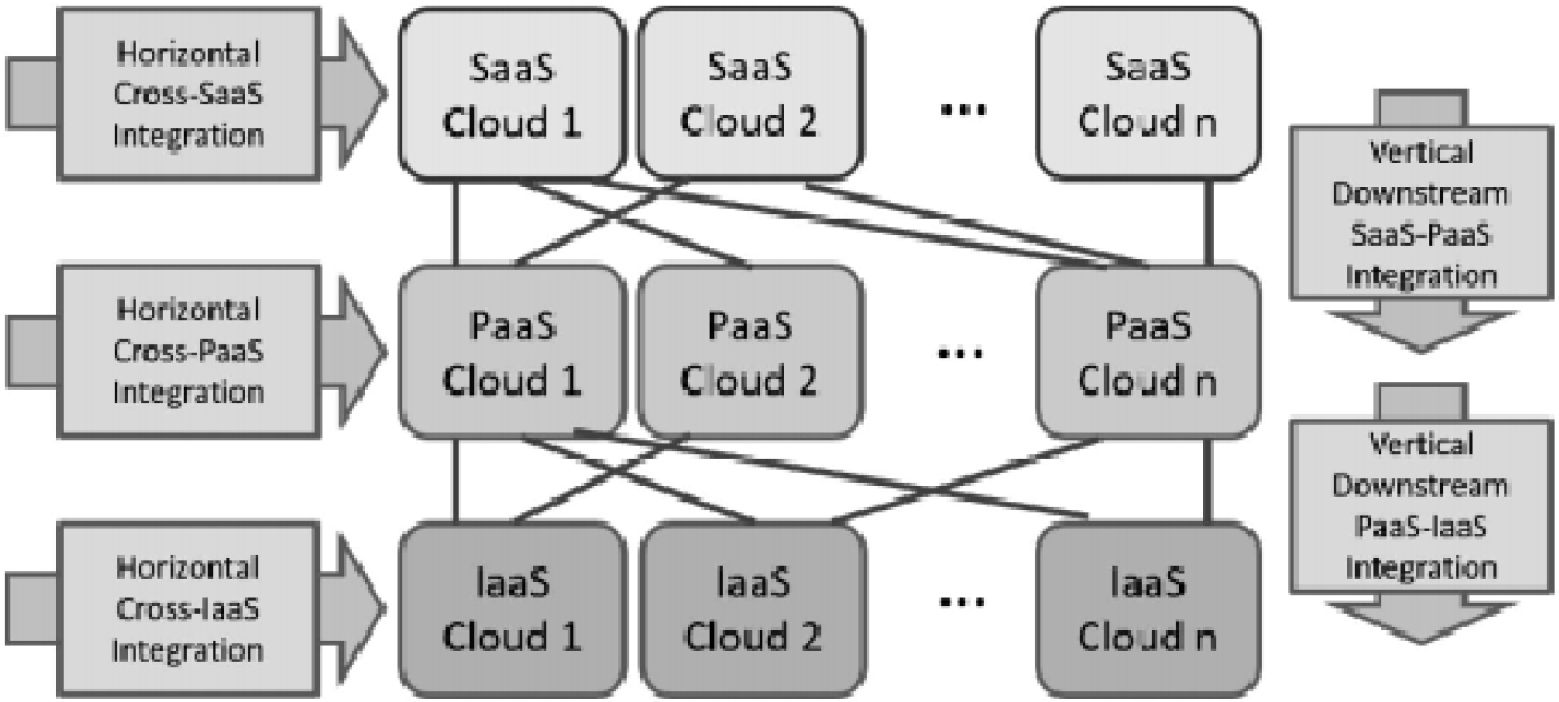



- X

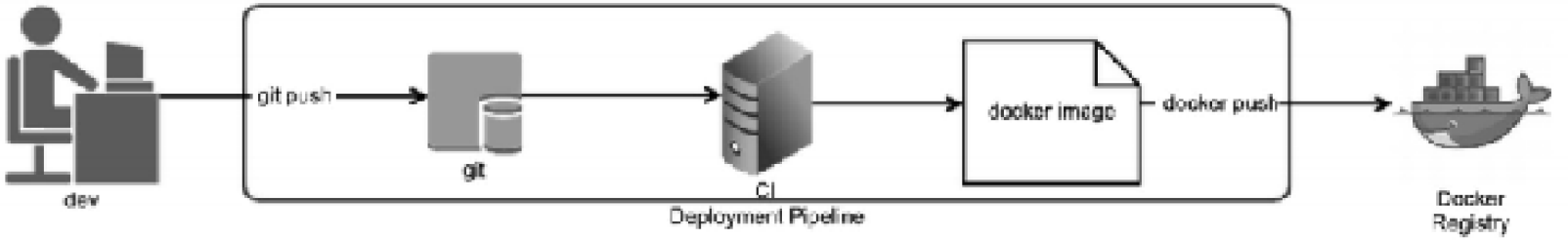
# Orchestrarea

- asigurarea resurse
- instanțele
- replanifcări
- relația cu interfața
- expunere servicii
- Mesos, Kubernetics, CorCos Tectonic si Docker Swarm

# Arhitectura containerelor în nor



# Dezvoltare continuă în Docker




Linie de productie (pipeline deployment) bazată pe containere

# Utilizarea Docker Compose

- definesc mediul
- definesc serviciile
- reguli suplimentare
- se lanseaza în execuție

docker-compose.yml

```
1.  version: '3'
2.  services:
3.  web:
4.  build: .
5.  ports:
6.  "5000:5000"
7.  volumes:
8.  .:/code
9.  - logvolume01:/var/log
10. inks:
11. - redis
12. redis:
13. image: redis
14. volumes:
```

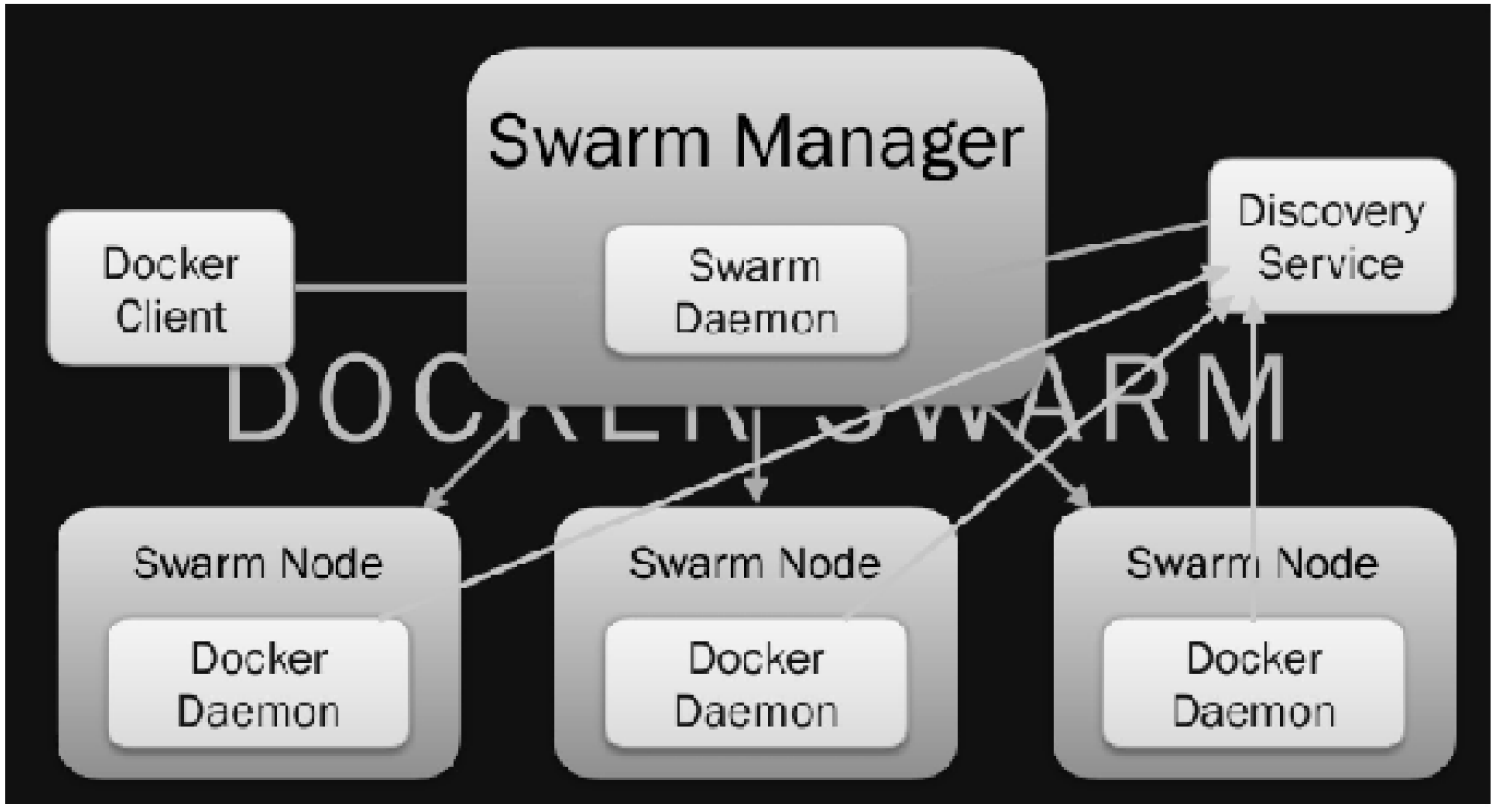
# Docker Compose

- Permite multiple medii de lucru izolate pe o singura masina gazda
- Pastreaza volumele de date in momentul crearii containerelor
- Recreaza numai containerele care s-au modificat
- Variabile si posibilitatatea schimbarii compozitiilor dintr-un mediu intr-altul

# Cand se foloseste Compose?

- dezvoltare
- testare automată
  - $ docker-compose up -d
  - $ ./run_tests
  - $ docker-compose down
- gazdă unică

# Orchestrarea containerelor




Docker Swarm

# Facilități oferite de Docker Swarm

- Management
- Scalarea
- Rețea între gazde
- Descoperirea serviciilor
- Echilibrarea încărcarii - Load balancing
- Nodul din roi

# Servicii și sarcini in Docker swarm

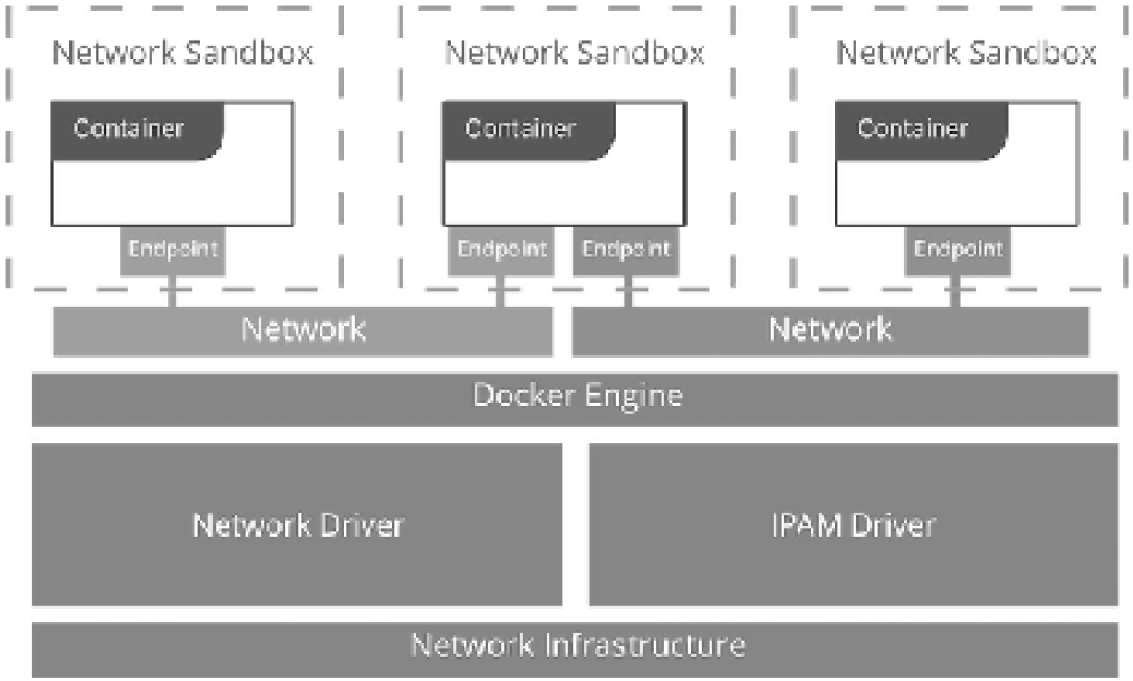
- serviciu
- sarcină/task
- planificare statică

# Echilibrarea încărcării în Docker swarm

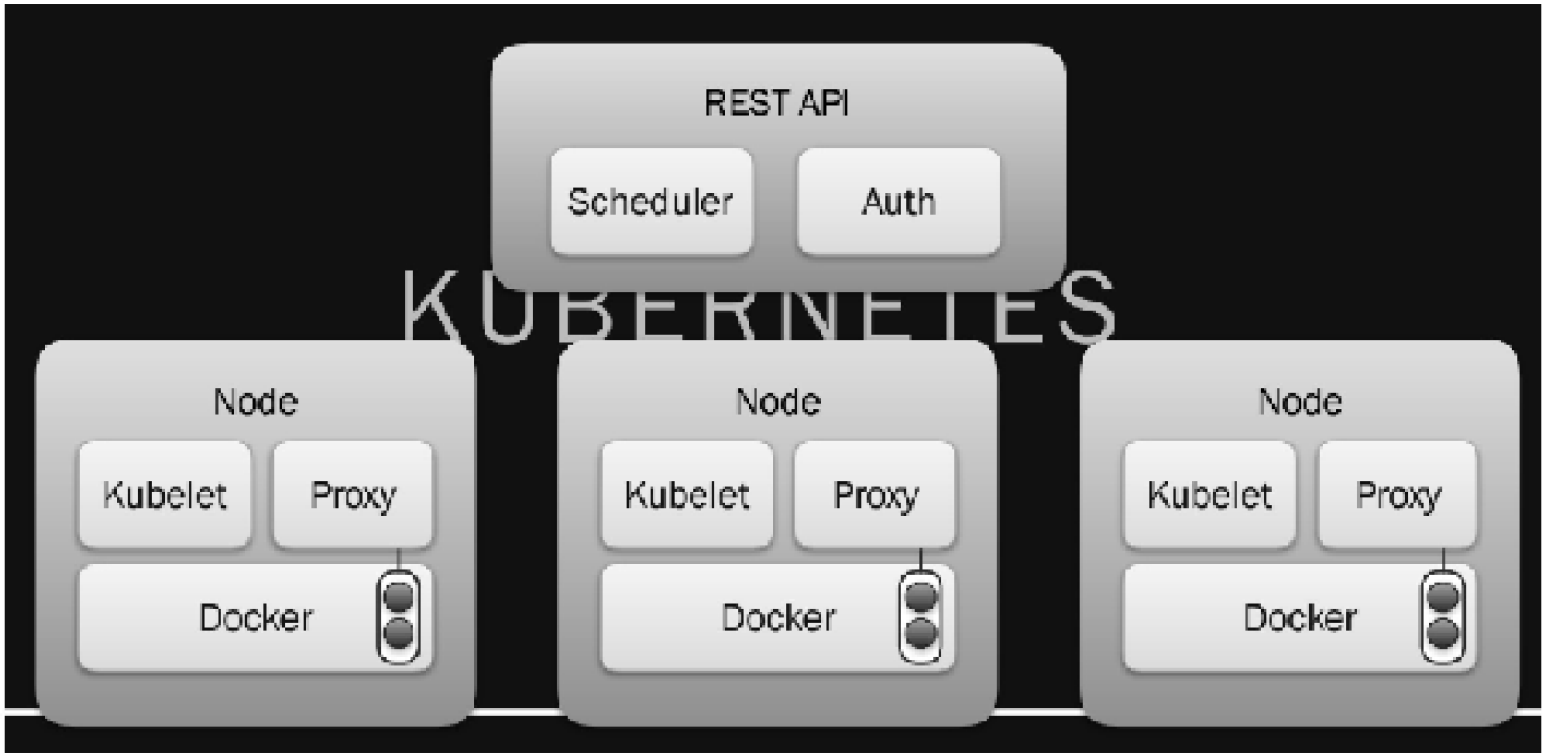
- port
  - manual
  - automat 30000-32767
- DNS local

# Filtrele Docker swarm

- Contrangere sau marcaj nod
- Afinitate
- Portul
- Dependența
- Sănătatea

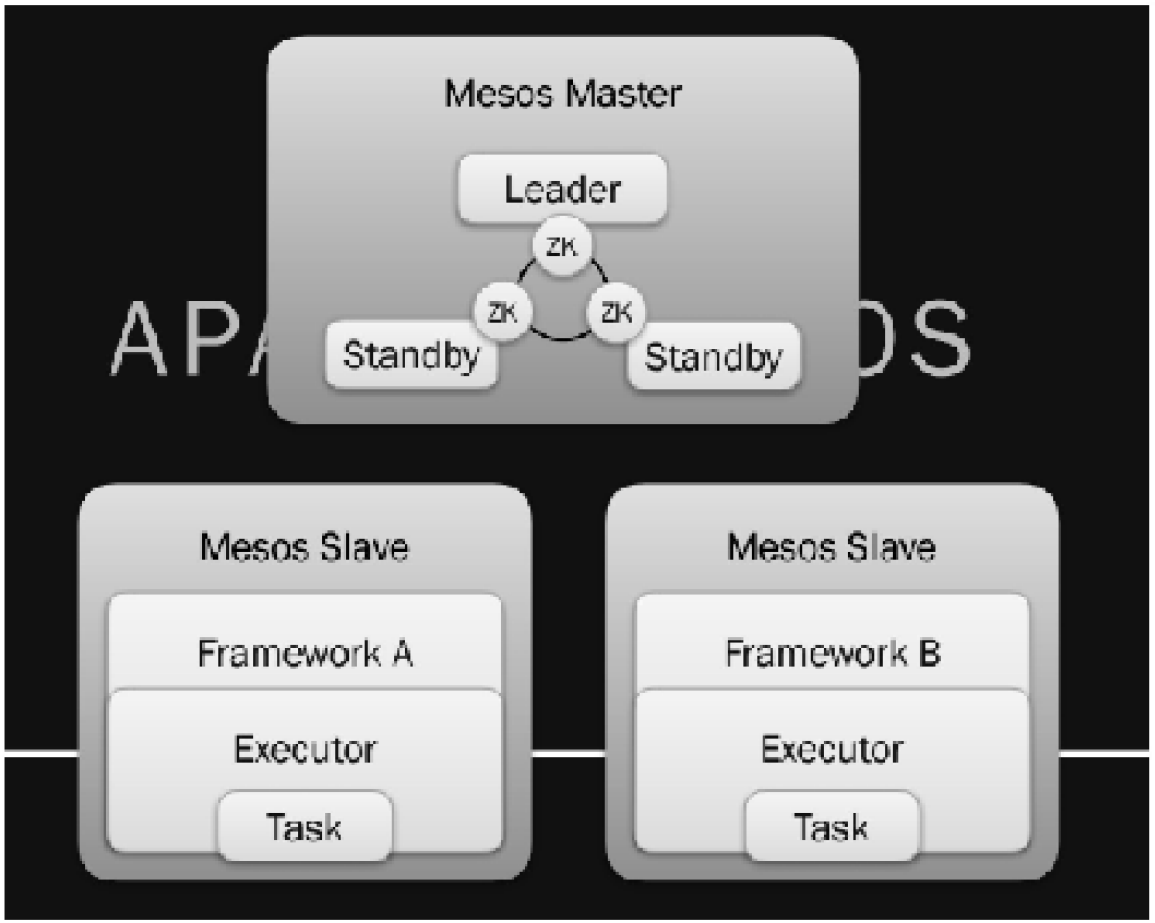


# Alte soluții pentru orchestrarea containerelor




Kubernetes

# Alte soluții pentru orchestrarea containerelor




Apache Mesos