Exercici d'alta disponibilitat amb Docker Swarm i Stack, però executat íntegrament en un únic node (entorn de laboratori o màquina local).

Nota conceptual important: En un sol node no hi ha alta disponibilitat real a nivell de node, però sí:

"Alta disponibilitat a nivell de servei"
"Replicació de contenidors"
"Recuperació automàtica de processos"
"Load balancing intern de Swarm"

Objectiu:
Simular un escenari d'alta disponibilitat en un sol host utilitzant:

Docker Swarm (single-node)
Docker Stack
Múltiples rèpliques
Reinici automàtic davant fallades

1. Inicialització del Swarm (un sol node)

bash> docker swarm init

Verificació:

bash> docker node ls

2. Fitxer docker-stack.yml yaml

```
services:
  web:
    image: nginx:alpine
    ports:
      - "8080:80"
    deploy:
      replicas: 3
      restart_policy:
        condition: on-failure
      update_config:
        parallelism: 1
        delay: 5s
    networks:
      - webnet

networks:
  webnet:
    driver: overlay
```

3. Desplegament de la Stack

bash> docker stack deploy -c docker-stack.yml webstack

Verificació:

bash>docker stack services webstack
bash>docker service ps webstack_web

4. Prova de càrrega i balanceig

bash>curl http://localhost:8080

Swarm distribueix les peticions entre les rèpliques (routing mesh).

5. Simulació de fallades

5.1 Aturar un contenidor manualment
bash> docker ps docker stop

Observació: El contenidor desapareix Swarm crea una nova rèplica automàticament

bash>docker service ps webstack_web

5.2 Fallada d'un procés dins el contenidor

bash>docker exec -it kill 1

Resultat: El contenidor mor Swarm el reinicia

6. Escalat dinàmic

bash>docker service scale webstack_web=5
Sense interrupció del servei.

7. Prova de rolling update Modificar la imatge: yaml image: nginx:latest

bash> docker stack deploy -c docker-stack.yml webstack

Swarm actualitza els contenidors un a un, sense downtime.