Contenidors per a l'administració de sistemes

Índex

Cicle formatiu: CFGS Administració de sistemes informàtics en xarxa (ASIX)

Mòdul: 0370 - Implantació de sistemes operatius / 0378 - Seguretat i alta disponibilitat

Figura 1: Docker logo

1. Què és Docker?

Docker és una plataforma de **contenidors** que permet empaquetar una aplicació amb totes les seves dependències (llibreries, binaris, configuració) en una unitat aïllada i portable anomenada **contenidor**. A diferència d'una màquina virtual, un contenidor no inclou un sistema operatiu sencer: comparteix el nucli (*kernel*) del sistema amfitrió i només aïlla els processos, la xarxa i el sistema de fitxers de l'aplicació.

Docker es va presentar el 2013 i està format per diversos components:

- **Docker Engine**: el dimoni (`dockerd`) que crea i gestiona els contenidors.
- **Docker CLI**: l'ordre `docker` que fem servir des del terminal.
- **containerd**: el *runtime* de baix nivell que gestiona el cicle de vida dels contenidors.
- **Docker Compose**: eina per definir i executar aplicacions multi-contenidor mitjançant un fitxer YAML.
- **Docker Hub**: registre públic d'imatges (`docker.io`).

1.1. Contenidors vs. màquines virtuals

Característica	Màquina virtual	Contenidor
Aïllament	Total (hipervisor + kernel propi)	Procés (namespaces + cgroups del kernel host)
Pes	GB (SO complet)	MB (només l'aplicació i dependències)
Arrencada	Minuts	Mil·lisegons/segons
Densitat	Baixa	Alta (desenes de contenidors per host)
Portabilitat	Depèn de l'hipervisor	Alta (mateixa imatge a qualsevol host amb Docker)

1.2. Per a què serveix?

- **Desplegament consistent**: “funciona a la meva màquina” deixa de ser un problema, perquè l'entorn viatja amb l'aplicació.
- **Microserveis**: cada servei (base de dades, backend, frontend, proxy) s'executa en el seu propi contenidor aïllat.
- **Entorns de desenvolupament i proves**: aixecar i destruir entorns complets en segons (útil per a laboratoris i pràctiques d'aula).
- **CI/CD**: integració en *pipelines* d'integració contínua per generar builds reproduïbles.
- **Aprofitament de recursos**: molta més densitat de serveis per màquina que amb VM tradicionals.

1.3. Conceptes clau

- **Imatge (*image*)**: plantilla de només lectura amb el codi, les dependències i la configuració. Es construeix per capes (*layers*).
- **Contenidor (*container*)**: instància en execució d'una imatge.
- **Dockerfile**: fitxer de text amb les instruccions per construir una imatge.
- **Volum (*volume*)**: mecanisme de persistència de dades gestionat per Docker, independent del cicle de vida del contenidor.
- **Xarxa (*network*)**: infraestructura virtual que permet la comunicació entre contenidors i amb l'exterior.
- **Registre (*registry*)**: dipòsit d'imatges (Docker Hub, GitHub Container Registry, registre privat propi, etc.).

2. Instal·lació a Ubuntu Server 26.04

Ubuntu inclou un paquet `docker.io` als seus dipòsits, però es recomana instal·lar Docker des del **dipòsit oficial de Docker** per obtenir la versió més recent i totes les eines (Buildx, Compose plugin, etc.).

2.1. Prepara el sistema

```
sudo apt update
sudo apt install ca-certificates curl gnupg
```

2.2. Desinstal·la versions antigues (si n'hi ha)

```
sudo apt remove docker docker-engine docker.io containerd runc
```

2.3. Afegeix el dipòsit oficial de Docker

```
# Crea el directori per a les claus GPG
sudo install -m 0755 -d /etc/apt/keyrings

# Descarrega i desa la clau GPG oficial
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
  -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Afegeix el dipòsit a les fonts d'APT
echo \
  "deb [arch=$(dpkg --print-architecture)
   ↪  signed-by=/etc/apt/keyrings/docker.asc] \
  https://download.docker.com/linux/ubuntu \
```

```
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update
```

NOTA

Si el dipòsit de Docker encara no publica paquets per al *codename* de la nova versió d'Ubuntu, es pot forçar temporalment l'última LTS suportada (per exemple `noble`) substituint la variable `VERSION_CODENAME` en la línia anterior.

2.4. Instal·la Docker Engine

```
sudo apt install docker-ce docker-ce-cli containerd.io \
  docker-buildx-plugin docker-compose-plugin
```

Paquets instal·lats:

- `docker-ce`: el dimoni Docker Engine (*Community Edition*).
- `docker-ce-cli`: la interfície de línia d'ordres.
- `containerd.io`: el *runtime* de contenidors.
- `docker-buildx-plugin`: construcció d'imatges multiplataforma.
- `docker-compose-plugin`: subordre `docker compose` (Compose v2, integrada al CLI).

2.5. Comprova la instal·lació

```
sudo systemctl status docker
sudo docker run hello-world
```

Si tot funciona correctament, es descarregarà la imatge `hello-world` i es mostrarà un missatge de confirmació.

2.6. Executa Docker sense sudo

Per defecte, calen privilegis d'administrador per interactuar amb el dimoni Docker (que escolta a través d'un *socket* Unix propietat de `root`). Per permetre que un usuari normal executi `docker` sense `sudo`:

```
sudo usermod -aG docker $USER
newgrp docker
```

AVÍS

Afegir un usuari al grup `docker` li dona, de facto, privilegis equivalents a `root` sobre el sistema, ja que es pot muntar qualsevol part del sistema de fitxers dins d'un contenidor. Cal tenir-ho present en entorns multiusuari.

Cal tancar sessió i tornar a entrar (o executar `newgrp docker`) perquè el canvi de grup tingui efecte.

3. Configuració bàsica del dimoni

3.1. Gestió del servei amb systemd

```
sudo systemctl start docker      # arrencar
sudo systemctl stop docker       # aturar
sudo systemctl restart docker    # reiniciar
sudo systemctl enable docker     # arrencada automàtica al boot
sudo systemctl disable docker    # desactivar arrencada automàtica
sudo systemctl status docker     # estat del servei
```

3.2. El fitxer `daemon.json`

La configuració global del dimoni es defineix a `/etc/docker/daemon.json` (no existeix per defecte; cal crear-lo).

```
sudo nano /etc/docker/daemon.json
```

Exemple de configuració habitual:

```
{
  "data-root": "/var/lib/docker",
  "storage-driver": "overlay2",
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "3"
  },
  "default-address-pools": [
    { "base": "172.30.0.0/16", "size": 24 }
  ],
  "insecure-registries": [],
  "registry-mirrors": [],
  "dns": ["8.8.8.8", "1.1.1.1"],
  "iptables": true,
  "live-restore": true
}
```

Després de modificar `daemon.json` cal reiniciar el dimoni:

```
sudo systemctl restart docker
```

3.3. Opcions de configuració més rellevants

Opció	Descripció
`data-root`	Directori on Docker desa imatges, contenidors i volums (per defecte `/var/lib/docker`). Útil per moure-ho a un disc amb més espai.
`storage-driver`	Controlador d'emmagatzematge de capes. `overlay2` és el recomanat en kernels moderns.
`log-driver` / `log-opts`	Controla com es guarden els *logs* dels contenidors. `json-file` amb rotació evita que els logs omplin el disc.
`default-address-pools`	Rangs d'IP que Docker assigna automàticament a les xarxes que es creïn.
`insecure-registries`	Llista de registres accessibles per HTTP (sense TLS) o amb certificat no vàlid, útil per a un registre intern del laboratori.
`registry-mirrors`	Mirall(s) del registre de Docker Hub, per accelerar descàrregues o evitar límits de *pull rate*.
`dns`	Servidors DNS que s'injecten dins dels contenidors.
`live-restore`	Permet que els contenidors segueixin en execució encara que es reiniciï el dimoni `dockerd`.
`default-runtime`	*Runtime* per defecte (`runc` és l'habitual; es pot canviar per `nvidia` amb el toolkit de GPU, per exemple).
`bip`	Adreça del pont (*bridge*) `docker0` per defecte.

3.4. Moure el directori de dades (`data-root`)

Si el disc del sistema és petit i es vol emmagatzemar les imatges en una altra partició:

```
sudo systemctl stop docker
sudo rsync -aP /var/lib/docker/ /nou/disc/docker/
```

A `daemon.json`:

```
{
  "data-root": "/nou/disc/docker"
}
```

```
sudo mv /var/lib/docker /var/lib/docker.old
sudo systemctl start docker
sudo docker info | grep "Docker Root Dir"
```

Un cop comprovat que funciona correctament, es pot eliminar `/var/lib/docker.old`.

3.5. Configuració de xarxa

Docker crea per defecte tres xarxes: `bridge`, `host` i `none`.

```
docker network ls
docker network inspect bridge
```

Crear una xarxa pròpia (recomanat per fer comunicar contenidors entre si per nom):

```
docker network create --driver bridge --subnet 172.28.0.0/16 xarxa_lab
```

3.6. Emmagatzematge persistent: volums

```
docker volume create dades_web
docker volume ls
docker volume inspect dades_web
```

Ús en executar un contenidor:

```
docker run -d --name web -v dades_web:/var/www/html nginx
```

3.7. Configuració de recursos (memòria i CPU)

Es pot limitar el consum de recursos per contenidor en el moment d'executar-lo:

```
docker run -d --name app \
  --memory="512m" \
  --cpus="1.5" \
  imatge_aplicacio
```

4. Ordres essencials

```
# Imatges
docker pull ubuntu:26.04          # descarregar una imatge
docker images                     # llistar imatges locals
docker rmi <imatge>                # eliminar una imatge
docker build -t app:1.0 .          # construir imatge des d'un
 ↪  Dockerfile

# Contenidors
docker ps                         # contenidors en execució
docker ps -a                      # tots els contenidors
docker run -d --name web -p 8080:80 nginx   # executar en segon pla
docker exec -it web bash          # obrir un shell dins del contenidor
```

```
docker logs -f web                # veure logs en temps real
docker stop web                   # aturar
docker start web                  # arrencar
docker rm web                     # eliminar

# Sistema
docker system df                  # espai utilitzat
docker system prune -a            # netejar imatges/contenidors no
 ↪  usats
docker info                       # informació general del dimoni
```

5. Docker Compose

Docker Compose permet definir aplicacions multi-contenidor en un fitxer `compose.yaml`:

```
services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - dades_web:/usr/share/nginx/html
    networks:
      - xarxa_lab

  db:
    image: mariadb:11
    environment:
      MARIADB_ROOT_PASSWORD: canvia_aquesta_contrasenya
    volumes:
      - dades_db:/var/lib/mysql
    networks:
      - xarxa_lab

volumes:
  dades_web:
  dades_db:

networks:
  xarxa_lab:
```

Ordres bàsiques:

```
docker compose up -d       # aixecar els serveis en segon pla
docker compose ps          # estat dels serveis
docker compose logs -f     # logs de tots els serveis
docker compose down        # aturar i eliminar els contenidors
```

6. Bones pràctiques per a un entorn de laboratori/aula

- Utilitzar sempre **volums amb nom** per a dades que han de persistir (bases de dades, contingut web), mai dependre del sistema de fitxers intern del contenidor.
- Definir **límits de recursos** (`--memory`, `--cpus`) quan es comparteix un servidor entre diversos alumnes o serveis.
- Activar la **rotació de logs** a `daemon.json` per evitar que el disc s'ompli.
- No exposar el *socket* de Docker (`/var/run/docker.sock`) dins de contenidors sense necessitat real: qui té accés al socket té control total del host.
- Fer servir **xarxes personalitzades** en lloc de la `bridge` per defecte, per aprofitar la resolució de noms interna entre contenidors.
- Revisar periòdicament `docker system df` i fer `docker system prune` per alliberar espai en disc.

7. Recursos addicionals

- Documentació oficial: https://docs.docker.com/
- Referència de `daemon.json`: https://docs.docker.com/engine/reference/commandline/dockerd/
- Docker Hub: https://hub.docker.com/

Versions d'aquest document

- HTML - docker.html
- PDF - docker.pdf
- ODT - docker.odt
- MD - docker.md