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

Actualitza la llista de paquets:

```
sudo apt update
```

Actualitza els paquets:

```
sudo apt upgrade
```

Instal·la dependències:

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

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

```
sudo apt remove docker 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
```

Canvia els permisos:

```
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
```

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.

Actualitza la llista de paquets:

```
sudo apt update
```

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

Verifica l'estat del servei:

```
sudo systemctl status docker
```

Sortida esperada:

```
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled;
 ↪  preset: enabled)
     Active: active (running) since Sun 2026-07-12 08:11:32 UTC; 4s ago
 Invocation: ed7c5a785e4f43309453bc1da198d2af
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 2410 (dockerd)
      Tasks: 9
     Memory: 32.2M (peak: 32.6M)
        CPU: 380ms
     CGroup: /system.slice/docker.service
             └─2410 /usr/bin/dockerd -H fd://
 ↪  --containerd=/run/containerd/containerd.sock

de jul. 12 08:11:31 server dockerd[2410]:
 ↪  time="2026-07-12T08:11:31.804386317Z" level=info msg="Restoring
 ↪  containers: start."
de jul. 12 08:11:31 server dockerd[2410]:
 ↪  time="2026-07-12T08:11:31.835503821Z" level=info msg="Deleting
 ↪  nftables IPv4 rules" error="running nft: /dev/stdin:1:17-30: E>
de jul. 12 08:11:31 server dockerd[2410]:
 ↪  time="2026-07-12T08:11:31.841391733Z" level=info msg="Deleting
 ↪  nftables IPv6 rules" error="running nft: /dev/stdin:1:18-31: E>
de jul. 12 08:11:32 server dockerd[2410]:
 ↪  time="2026-07-12T08:11:32.043110112Z" level=info msg="Loading
 ↪  containers: done."
de jul. 12 08:11:32 server dockerd[2410]:
 ↪  time="2026-07-12T08:11:32.048096969Z" level=info msg="Docker
 ↪  daemon" commit=8ec5ab3 containerd-snapshotter=true storage-drive>
de jul. 12 08:11:32 server dockerd[2410]:
 ↪  time="2026-07-12T08:11:32.048275170Z" level=info
 ↪  msg="Initializing buildkit"
de jul. 12 08:11:32 server dockerd[2410]:
 ↪  time="2026-07-12T08:11:32.279767556Z" level=info msg="Completed
 ↪  buildkit initialization"
de jul. 12 08:11:32 server dockerd[2410]:
 ↪  time="2026-07-12T08:11:32.288558516Z" level=info msg="Daemon has
 ↪  completed initialization"
de jul. 12 08:11:32 server dockerd[2410]:
 ↪  time="2026-07-12T08:11:32.288759223Z" level=info msg="API listen
 ↪  on /run/docker.sock"
de jul. 12 08:11:32 server systemd[1]: Started docker.service - Docker
 ↪  Application Container Engine.
```

Engega la imatge `hello-world`:

```
sudo docker run hello-world
```

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

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
4f55086f7dd0: Pull complete
d5e71e642bf5: Download complete
Digest: sha256:96498ffd522e70807ab6384a5c0485a79b9c7c08ca79ba08623ed
 ↪  cad1054e62d
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working
 ↪  correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker
 ↪  Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which
 ↪  runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which
 ↪  sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

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
```

Aplica els canvis:

```
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      # arrenca
sudo systemctl stop docker       # atura
sudo systemctl restart docker    # reinicia
sudo systemctl enable docker     # arrencada automàtica al boot
sudo systemctl disable docker    # desactiva l'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 reinicïi 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`.

Llista totes les xarxes Docker existents al host:

```
sudo docker network ls
```

```
NETWORK ID     NAME      DRIVER    SCOPE
53c0cedf1824   bridge    bridge    local
ab66adc4d25b   host      host      local
a46fd04d0513   none      null      local
```

Mostra informació detallada (en format JSON) sobre la xarxa `bridge`, que és la xarxa per defecte que fan servir els contenidors quan no n'especifiques cap altra.

- **Subnet/Gateway:** el rang d'IP que fa servir aquesta xarxa i la porta d'enllaç.
- **Containers:** quins contenidors estan connectats a aquesta xarxa ara mateix, amb la seva IP interna assignada.
- **Driver/Options:** configuració del driver de xarxa (MTU, icc, etc.).

És molt útil quan vols saber, per exemple, quina IP té un contenidor concret dins la xarxa `bridge` per poder-hi connectar des d'un altre contenidor o des del host.

```
sudo docker network inspect bridge
```

```
[
    {
        "Name": "bridge",
        "Id": "53c0cedf182403d3d7edcdac83be248352cdbaade0dd1ec09a6562
↪  5fcdda1f9f",
        "Created": "2026-07-12T08:14:55.849538694Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
```

```
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {},
        "Containers": {},
        "Status": {
            "IPAM": {
                "Subnets": {
                    "172.17.0.0/16": {
                        "IPsInUse": 3,
                        "DynamicIPsAvailable": 65533
                    }
                }
            }
        }
    }
]
```


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

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


```
83e4c5ae9802e8f1c543b0dbe9609e1fe99aa4eaa569306883fcb040cb77837f
```


Desglossament:

- `sudo docker network create` → crea una nova xarxa Docker.
- `--driver bridge` → fa servir el driver **bridge**, que és el més habitual per a xarxes locals en un sol host. Crea una xarxa privada aïllada (similar a un switch virtual) a la qual es poden connectar contenidors.
- `--subnet 172.28.0.0/16` → defineix el rang d'adreces IP que Docker

assignarà als contenidors connectats a aquesta xarxa (des de `172.28.0.1` fins a `172.28.255.254`, ja que és una màscara /16).
- `xarxa_lab` → el nom que li poses a la xarxa.

Què aconsegueixes amb això:

- Es crea un bridge Linux nou (visible amb `ip a` o `brctl show`) diferent del bridge per defecte de Docker (`docker0`).
- Els contenidors que hi connectis (amb `docker run --network xarxa_lab ...` o des d'un `docker-compose.yml`) rebran IP dins de `172.28.0.0/16` i podran comunicar-se entre ells per nom de contenidor (Docker fa resolució DNS interna automàtica en xarxes bridge personalitzades, cosa que **no** passa a la xarxa `bridge` per defecte).
- Queden aïllats d'altres xarxes/contenidors que no hi estiguin connectats explícitament, cosa útil per simular un laboratori de xarxes separat (per exemple, per practicar DHCP, DNS, o segmentació sense interferir amb altres serveis del host).

CONSELL

Et pot ser útil si vols simular una xarxa aïllada per a proves abans de tocar la infraestructura "real".

3.6. Emmagatzematge persistent: volums

Crea un volum Docker anomenat `dades_web`:

```
sudo docker volume create dades_web
```

Per defecte, quan un contenidor s'esborra, totes les dades que hi havia dins també desapareixen. Els volums resolen això: es guarden **fora** del sistema de fitxers del contenidor (normalment a `/var/lib/docker/volumes/dades_web/_data` al host), de manera que:

- Les dades sobreviuen encara que esborris i tornis a crear el contenidor.
- Es poden compartir entre diversos contenidors alhora.
- Docker se n'encarrega de gestionar-los (a diferència dels *bind mounts*, on tu apuntes directament a una carpeta concreta del host).

Llista els volums existents:

```
sudo docker volume ls
```

Mostra detalls (ubicació al host, driver, etc.):

```
sudo docker volume inspect dades_web
```

Ús en executar un contenidor:

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

Aquí, tot el que el contenidor `web` escrigui a `/var/www/html` es guardarà dins del volum `dades_web`, i sobreviurà encara que el contenidor mori.

CONSELL

Útil, per exemple, si vols persistir la base de dades d'un WordPress o Moodle en un contenidor, sense perdre-la cada cop que actualitzes la imatge.

3.7. Configuració de recursos (memòria i CPU)

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

```
sudo 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/

Índex de figures

Versions d'aquest document

- HTML
- PDF
- ODT
- MD