Generalitat de Catalunya
Departament d'Educació
Institut Thos i Codina

Docker Compose

Domini Públic (CC0)

Índex

1. Per a què fer servir Docker Compose? **1**

2. Funcionalitats principals **2**

3. Instal·lació a Ubuntu 26.04 **2**
3.1. Mètode recomanat: repositori oficial de Docker 3
3.2. Mètode alternatiu: paquet dels repositoris d'Ubuntu 4

4. Configuració: el fitxer `compose.yaml` **5**
4.1. Estructura bàsica 5
4.2. Fitxer `.env` 6
4.3. Ordres bàsiques d'ús 6
4.4. Múltiples fitxers per a diferents entorns 7

5. Bones pràctiques **7**

6. Referències a la documentació oficial **7**

Índex de figures **8**

Docker Compose és una eina oficial de Docker que permet definir i executar aplicacions formades per **múltiples contenidors** a partir d'un únic fitxer de configuració en format YAML (`compose.yaml`, tradicionalment anomenat `docker-compose.yml`).

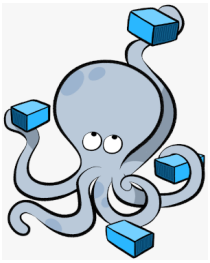

Figura 1: Docker Compose logo

En lloc d'haver d'executar diverses ordres `docker run` amb els seus paràmetres de xarxa, volums i variables d'entorn, es defineix tota l'arquitectura de l'aplicació (serveis, xarxes, volums) en un sol fitxer declaratiu i es gestiona amb un únic client: `docker compose`.

NOTA

Des de la versió 2 (Compose V2), l'eina es distribueix normalment com un **connector (plugin)** del client de Docker, invocat com `docker compose` (sense guionet), i ja no com el binari independent `docker-compose`. El binari clàssic `docker-compose` (V1) està en desús i no rep actualitzacions des de juliol de 2023.

1. Per a què fer servir Docker Compose?

- **Reproductibilitat**: tota la infraestructura queda descrita en un fitxer versionable amb Git.
- **Simplicitat**: una sola ordre (`docker compose up`) aixeca tots els serveis interconnectats.
- **Entorns aïllats**: cada projecte pot tenir la seva pròpia xarxa i volums, sense interferir amb altres.
- **Escalabilitat de desenvolupament**: molt útil per a entorns de laboratori, proves i desenvolupament (menys recomanat, per si sol, per a producció a gran escala, on se solen usar orquestradors: Docker Swarm, que accepta fitxers Compose amb `docker stack deploy`, o Kubernetes, al qual normalment es migra mitjançant eines de conversió com Kompose).

2. Funcionalitats principals

Funcionalitat	Descripció
Definició declarativa de serveis	Cada contenidor de l'aplicació es defineix com un `service` amb la seva imatge, ports, variables d'entorn, etc.
Xarxes automàtiques	Compose crea per defecte una xarxa pont (*bridge*) pròpia del projecte perquè els serveis es puguin resoldre pel seu nom (DNS intern).
Gestió de volums	Permet definir volums amb nom o *bind mounts* per a la persistència de dades.
Variables d'entorn i fitxers `.env`	Es poden parametritzar els valors del `compose.yaml` mitjançant un fitxer `.env` o variables d'entorn del sistema.
Dependències entre serveis	La clau `depends_on` permet controlar l'ordre d'arrencada (i, amb `condition`, esperar que un servei estigui *healthy*).
Escalat de serveis	`docker compose up --scale <servei>=N` permet aixecar diverses rèpliques d'un mateix servei.
Perfils (`profiles`)	Permet activar o desactivar grups de serveis segons el context (p. ex. `debug`, `testing`).
Build integrat	Compose pot construir imatges pròpies (`build:`) a partir d'un `Dockerfile`, en lloc de descarregar-les d'un registre.
Múltiples fitxers de composició	Es poden combinar diversos fitxers (`-f base.yaml -f override.yaml`) per a diferents entorns (dev, prod).
Healthchecks	Definició de comprovacions d'estat (`healthcheck`) per a cada servei.
Gestió del cicle de vida complet	Ordres per crear, aturar, reiniciar, escalar, veure registres i eliminar tota la pila amb una sola eina.

3. Instal·lació a Ubuntu 26.04

NOTA

Docker Compose V2 s'instal·la com a **connector** (`docker-compose-plugin`) del client de Docker, integrat amb l'ordre `docker`. Es recomana instal·lar-lo des del repositori oficial de Docker per obtenir sempre l'última versió estable, en lloc del paquet `docker-compose-v2` dels repositoris d'Ubuntu (més estable però normalment amb versions més antigues).

3.1. Mètode recomanat: repositori oficial de Docker

Pas 1. Elimina paquets conflictius

Si el sistema té paquets antics de Docker o Podman instal·lats des dels repositoris d'Ubuntu, cal eliminar-los primer:

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 \
           podman-docker containerd runc; do
    sudo apt remove -y "$pkg" 2>/dev/null
done
```

NOTA

És normal que `apt` informi que alguns d'aquests paquets no estan instal·lats; no és cap error.

Pas 2. Actualitza l'índex de paquets i instal·lar dependències

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

Pas 3. Afegeix la clau GPG oficial de Docker

```
sudo install -m 0755 -d /etc/apt/keyrings
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
```

Pas 4. Afegeix el repositori d'APT de Docker

```
sudo tee /etc/apt/sources.list.d/docker.sources > /dev/null <<EOF
Types: deb
URIs: https://download.docker.com/linux/ubuntu
Suites: $(. /etc/os-release && echo
 ↪  "${UBUNTU_CODENAME:-$VERSION_CODENAME}")
Components: stable
Architectures: $(dpkg --print-architecture)
Signed-By: /etc/apt/keyrings/docker.asc
EOF

sudo apt update
```

NOTA

Aquest és el format *deb822* (fitxer `.sources`) que fa servir actualment la documentació oficial de Docker. El format antic d'una sola línia (`docker.list`) encara funciona, però ja no és el recomanat. L'expressió `${UBUNTU_CODENAME:-$VERSION_CODENAME}` fa que l'ordre també funcioni en derivades d'Ubuntu (p. ex. Linux Mint).

Pas 5. Instal·la Docker Engine i el connector Compose

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

Aquesta ordre instal·la conjuntament:

- `docker-ce` i `docker-ce-cli`: el motor de Docker i el seu client.
- `containerd.io`: el *runtime* de contenidors.
- `docker-buildx-plugin`: constructor d'imatges avançat (multiplataforma).
- `docker-compose-plugin`: el connector `docker compose` (Compose V2).

Pas 6. Verifica la instal·lació

```
docker compose version
```

Hauria de mostrar alguna cosa semblant a:

```
Docker Compose version v5.x.x
```

Pas 7. Permet l'ús de Docker sense sudo (opcional però recomanat)

```
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 (el dimoni de Docker s'executa com a `root`). Cal valorar-ho amb cura en entorns multiusuari o de laboratori compartit.

Pas 8. Comprova que el dimoni funciona

```
sudo systemctl enable --now docker
docker run hello-world
```

3.2. Mètode alternatiu: paquet dels repositoris d'Ubuntu

Per a un entorn més senzill (encara que amb versions potencialment més antigues), Ubuntu 26.04 ofereix el paquet `docker-compose-v2`, que instal·la també Docker Engine (`docker.io`) com a dependència:

```
sudo apt update
sudo apt install -y docker-compose-v2
sudo usermod -aG docker "$USER"
newgrp docker
```

4. Configuració: el fitxer `compose.yaml`

4.1. Estructura bàsica

```
services:
  web:
    image: nginx:1.30
    ports:
      - "8080:80"
    volumes:
      - ./html:/usr/share/nginx/html:ro
    depends_on:
      db:
        condition: service_healthy

  db:
    image: mariadb:11
    environment:
      MARIADB_ROOT_PASSWORD: exemple
      MARIADB_DATABASE: appdb
    volumes:
      - db_data:/var/lib/mysql
    healthcheck:
      test: ["CMD", "healthcheck.sh", "--connect",
      ↪  "--innodb_initialized"]
      interval: 10s
      retries: 5

volumes:
  db_data:
```

Elements clau del fitxer:

- **`services`**: cada clau és el nom d'un servei/contenidor.
  - `image`: imatge a utilitzar (o `build:` per construir-la des d'un `Dockerfile`).
  - `ports`: mapatge `host:contenidor`.
  - `volumes`: muntatges persistents o *bind mounts*.
  - `environment`: variables d'entorn del contenidor.
  - `depends_on`: ordre d'arrencada entre serveis (amb `condition: service_healthy` espera que el servei del qual depengui superi el seu `healthcheck`).
  - `healthcheck`: comprovació d'estat del servei.
- **`volumes`** (nivell arrel): declaració dels volums amb nom gestionats per Docker.
- **`networks`** (nivell arrel, opcional): xarxes personalitzades, si no es vol la xarxa per defecte.

4.2. Fitxer `.env`

Es pot crear un fitxer `.env` al mateix directori per parametritzar valors:

```
# .env
DB_PASSWORD=canvia_aquesta_clau
WEB_PORT=8080
```

I referenciar-los al `compose.yaml`:

```
services:
  web:
    ports:
      - "${WEB_PORT}:80"
  db:
    environment:
      MARIADB_ROOT_PASSWORD: ${DB_PASSWORD}
```

AVÍS

No pugeu mai el fitxer `.env` amb credencials reals a un repositori Git públic. Afegiu-lo a `.gitignore` i distribuïu, si cal, un `.env.example` sense valors sensibles.

4.3. Ordres bàsiques d'ús

Ordre	Funció
`docker compose up -d`	Crea i arrenca tots els serveis en segon pla (*detached*).
`docker compose down`	Atura i elimina els contenidors, xarxes (i, amb `-v`, també els volums).
`docker compose ps`	Llista l'estat dels serveis del projecte.
`docker compose logs -f web`	Mostra els registres en temps real d'un servei concret.
`docker compose exec web bash`	Obre una consola dins d'un contenidor en execució.
`docker compose build`	Reconstrueix les imatges definides amb `build:`.
`docker compose config`	Valida i mostra el fitxer de composició resolt (útil per depurar).
`docker compose restart web`	Reinicia un servei concret.
`docker compose up --scale web=3 -d`	Escala el servei `web` a 3 rèpliques (vegeu l'avís següent).

AVÍS

`--scale` falla si el servei publica un port fix de l'amfitrió (p. ex. `"8080:80"` de l'exemple anterior), perquè les rèpliques no poden ocupar el mateix port. Per escalar cal publicar només el port del contenidor (`"80"`), usar un rang de ports (`"8080-8082:80"`) o posar un proxy invers al davant.

4.4. Múltiples fitxers per a diferents entorns

```
docker compose -f compose.yaml -f compose.prod.yaml up -d
```

Aquest patró permet mantenir un `compose.yaml` base i sobreescriure'n valors concrets (ports, rèpliques, variables) segons l'entorn (desenvolupament, producció, laboratori) sense duplicar tota la configuració.

5. Bones pràctiques

- Utilitzar **volums amb nom** per a dades que han de persistir (bases de dades), i **bind mounts** només per a codi font o fitxers de configuració que es volen editar des de l'amfitrió.
- Fixar sempre una **etiqueta de versió** de la imatge (`mariadb:11`) en lloc de `latest`, per garantir reproductibilitat.
- Definir `healthcheck` als serveis dels quals depenen altres serveis, i utilitzar `depends_on: condition: service_healthy` per esperar-los correctament.
- Separar secrets i contrasenyes en fitxers `.env` exclosos del control de versions.
- Validar el fitxer amb `docker compose config` abans de desplegar canvis importants.

6. Referències a la documentació oficial

- Documentació general de Compose: https://docs.docker.com/compose/
- Referència completa del fitxer Compose (especificació): https://docs.docker.com/reference/compose-file/
- Referència de l'ordre `docker compose`: https://docs.docker.com/reference/cli/docker/compose/
- Instal·lació de Docker Engine a Ubuntu: https://docs.docker.com/engine/install/ubuntu/
- Repositori oficial del projecte Compose (GitHub): https://github.com/docker/compose
- Especificació oberta del format Compose (Compose Specification): https://github.com/compose-spec/compose-spec

Índex de figures

1 Docker Compose logo 1

Versions d'aquest document

- HTML
- PDF
- ODT
- MD

Domini Públic (CC0)