Capitolo 6 – Docker Compose

In questo capitolo vedremo:

- Cos’è Docker Compose
- Installazione del plugin Compose
- Struttura di un file docker-compose.yml
- Gestione di applicazioni multi-container
- Comandi principali
- Variabili d’ambiente
- Best practice

6.1 Cos’è Docker Compose

Docker Compose è uno strumento che permette di definire e gestire applicazioni multi-container tramite un file YAML.

Con un solo comando è possibile:

- Creare reti
- Creare volumi
- Avviare più container
- Gestire dipendenze tra servizi

È ideale per ambienti di sviluppo e piccole/medie infrastrutture.

6.2 Verifica installazione

Docker Compose è incluso come plugin nelle versioni recenti di Docker.

Verifica:

```
docker compose version
```

Se il comando risponde con una versione, Compose è correttamente installato.

6.3 Struttura base di docker-compose.yml

Esempio semplice con Nginx:

```
services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
```

Salvare il file come:

```
docker-compose.yml
```

6.4 Avviare un progetto Compose

Dalla directory dove si trova il file:

```
docker compose up -d
```

Parametri:

- `up` → crea e avvia i servizi
- `-d` → modalità detached (background)

Verifica container attivi:

```
docker compose ps
```

6.5 Fermare e rimuovere i servizi

Fermare i container:

```
docker compose stop
```

Fermare e rimuovere container, reti e configurazioni:

```
docker compose down
```

Rimuovere anche i volumi:

```
docker compose down -v
```

6.6 Esempio applicazione multi-container (Web + Database)

Esempio con Nginx e MySQL:

```
services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
    depends_on:
      - db

  db:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: esempio
    volumes:
      - db_data:/var/lib/mysql

volumes:
  db_data:
```

In questo esempio:

- Viene creata automaticamente una rete
- I servizi possono comunicare usando il nome del servizio (es. `db`)
- Il volume `db_data` salva i dati del database

6.7 Variabili d’ambiente con file .env

Creare un file `.env` nella stessa directory:

```
MYSQL_ROOT_PASSWORD=superpassword
```

Nel file docker-compose.yml:

```
environment:
  MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
```

Compose caricherà automaticamente le variabili dal file `.env`.

6.8 Ricostruire i servizi

Se modifichi il Dockerfile:

```
docker compose up -d --build
```

Forzare la ricreazione dei container:

```
docker compose up -d --force-recreate
```

6.9 Visualizzare i log

Log di tutti i servizi:

```
docker compose logs
```

Seguire i log in tempo reale:

```
docker compose logs -f
```

Log di un singolo servizio:

```
docker compose logs web
```

6.10 Scalare un servizio

Esempio:

```
docker compose up -d --scale web=3
```

Questo avvia 3 istanze del servizio web.

Best Practice

- Non usare `latest` in produzione
- Separare ambienti (dev, staging, prod)
- Usare file `.env` per password e configurazioni
- Versionare sempre il file docker-compose.yml
- Utilizzare volumi per dati persistenti
- Limitare CPU e RAM nei servizi critici

Conclusione

Ora sai:

- Cos’è Docker Compose
- Gestire applicazioni multi-container
- Usare variabili d’ambiente
- Scalare servizi

- Gestire reti e volumi automaticamente

Nel prossimo capitolo vedremo come mettere in produzione un'applicazione Docker con reverse proxy e HTTPS.

Revision #1
Created 2026-02-27 16:40:04 UTC by Pe
Updated 2026-02-27 16:41:14 UTC by Pe