

Mini guida Docker – Avvio, Debug e Test Microservizi

Buildare / ricreare le immagini

Ricostruisce le immagini senza cache

```
docker-compose build --no-cache
```

Avvia nuovamente i container

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

Utile quando:

- cambi codice;
 - modifichi Dockerfile;
 - hai errori strani persistenti.
-

Avviare i container

Posizionati nella cartella del progetto (dove si trova docker-compose.yml)

```
cd C:\Users\decao\Desktop\Docker_Example (o semplicemente CMD)
```

Avvia tutti i servizi definiti in docker-compose

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

- up → crea e avvia i container;
- -d → esecuzione in background (detached).

Verifica che siano attivi:

```
docker ps
```

MINI GUIDA PER DOCKER

Fermare e rimuovere i container

```
# Ferma e rimuove i container
```

```
docker-compose down
```

Opzione **pulizia completa** (utile in caso di problemi persistenti):

```
# Rimuove anche volumi (DB) e immagini
```

```
docker-compose down -v --rmi all
```

- `-v` → elimina i volumi (attenzione: cancella i dati DB)
 - `--rmi all` → rimuove le immagini buildate
-

Controllare i log

```
# Log di tutti i container
```

```
docker-compose logs -f
```

```
# Log di un container specifico (es. orders)
```

```
docker logs -f docker_example-orders-1
```

- `-f` → segue i log in tempo reale;
 - fondamentale per capire **perché un servizio va in down**.
-

Testare le API con PowerShell (iwr)

Gateway – Users

```
iwr http://localhost:4000/api/users
```

Gateway – Orders

```
iwr http://localhost:4000/api/orders
```

Creare un nuovo ordine (POST)

```
iwr -Method POST http://localhost:4000/api/orders  
-Body (@{ item="Keyboard"; quantity=2 } | ConvertTo-Json);  
-ContentType "application/json".
```

Senza POST e senza Body, **non vengono creati dati** nel database.

SEZIONE DEBUG – Errore riscontrato in merito al database (crash o errore)

Errore

{"error":"Orders service unavailable"}

e nei test diretti su PostgreSQL:

FATAL: role "postgres" does not exist

FATAL: role "root" does not exist

Causa

- Il servizio Orders tentava di collegarsi con un **utente errato**
 - Il database era avviato, ma:
 - utente sbagliato
 - tabella orders assente
 - Risultato: **Orders va in crash quando riceve una richiesta**
-

MINI GUIDA PER DOCKER

Soluzione

Scoprire le credenziali corrette del DB

```
docker inspect docker_example-db-orders-1
```

Questo comando fornisce anche variabili d'ambiente quali:

- POSTGRES_USER;
- POSTGRES_DB;
- POSTGRES_PASSWORD.

Serve per capire il nome utente e database (sono sotto la sezione ENV).

Attivare correttamente a PostgreSQL

```
docker exec -it docker_example-db-orders-1 psql -U orders -d ordersdb
```

- docker exec -it → entra nel container;
- psql → client PostgreSQL;
- -U orders → utente corretto;
- -d ordersdb → database corretto.

L'uso di psql senza parametri porta all'errore 'role does not exist'.

(qui sono stati usare orders e ordersdb perché valori trovati con il comando precedente)

Verificare le tabelle esistenti

```
\dt
```

- mostra le tabelle del database

(comanda dato dopo il precedente perché entro all'interno di psql)

MINI GUIDA PER DOCKER

Creare la tabella Orders (se necessario altrimenti uso un'automazione)

```
CREATE TABLE orders (
    id SERIAL PRIMARY KEY,
    item VARCHAR(255) NOT NULL,
    quantity INTEGER NOT NULL
);
```

Verifica:

```
\dt
```

(comando da dare sempre da dentro psql)

Creare un'automazione per la tabella Orders

Creo un folder nella cartella del progetto (e.g. lo chiamo ‘db’), all’interno ci metto un file

- init.sql - Contiene i comandi per creare la tabella orders (posso metterne anche altri).

Modifico il file ‘docker-compose.yml’ nella sezione ‘db-orders’ aggiungo sotto la voce ‘volumes’

- - ./orders-service/db/init.sql:/docker-entrypoint-initdb.d/init.sql

Spiegazione:

- docker-entrypoint-initdb.d → PostgreSQL esegue automaticamente i .sql;
 - niente più accesso manuale per creare tabelle.
-

MINI GUIDA PER DOCKER

Riavviare il servizio Orders

```
docker restart docker_example-orders-1
```

(riavvio il singolo componente)

Controllo log:

```
docker logs docker_example-orders-1
```

Risultato corretto atteso:

Orders service running on port 4002

Possibile Q.A.:

Perché senza POST non vedi dati (giustamente)?

- GET /api/orders:
 - legge dal database;
 - se la tabella è vuota → restituisce [].
 - POST /api/orders:
 - inserisce dati nel database;
 - solo dopo il POST il GET restituisce risultati.
-

Stato finale corretto

- **Gateway:** attivo
- **Orders service:** stabile
- **Database:** connesso
- **GET /api/orders:** funzionante
- **POST /api/orders:** inserisce dati

Verifica:

```
iwr http://localhost:4000/
```

```
iwr http://localhost:4000/api/users
```

```
iwr http://localhost:4000/api/orders
```

```
iwr -Method POST http://localhost:4000/api/orders -Body (@{  
item="Keyboard"; quantity=2 } | ConvertTo-Json) -ContentType  
"application/json"
```
