GABRIELE ROMANATO

Come funzionano i controlli di integrità in Docker Compose

In questo articolo esamineremo nel dettaglio come possiamo gestire i controlli di integrità con Docker Compose.

I controlli di integrità sono un modo efficace per gestire i container all'interno di un contesto Docker Compose. Quando un container dipende da un altro container per funzionare correttamente, solitamente specifichiamo una regola `depends_on` nel blocco del container di destinazione.

Tuttavia, questa regola non implica che Docker eseguirà un controllo sullo stato del container specificato in questa regola. Significa invece che tale container verrà avviato prima del container di destinazione. In altre parole, è solo un modo per definire un ordine sequenziale nell'avvio dei container.

Supponiamo che la nostra app dipenda da un database e che il database debba essere pronto ad accettare connessioni dalla nostra app. Ecco un esempio di come possiamo utilizzare i controlli di integrità in questo scenario:

```
db:
  image: postgres
  container_name: db
  environment:
    POSTGRES_USER: ${DB_USER}
    POSTGRES_PASSWORD: ${DB_PASSWORD}
    POSTGRES_DB: ${DB_NAME}
  ports:
    - "127.0.0.1:5432:5432"
  volumes:
    - dbdata:/var/lib/postgresql/data
  healthcheck:
    test: ["CMD", "pg_isready", "-U", "${DB_USER}"]
    interval: 10s
```

```
      timeout: 5s
      retries: 5
  app:
    depends_on:
      db:
        condition: service_healthy
```

Il blocco `healthcheck` ha le seguenti proprietà:

1. `test`: Il comando shell che esegue il test. In questo caso specifico, l'utilità da riga di comando utilizzata è incorporata, ma in molti altri casi, ad esempio se è necessario usare `curl`, lo strumento a riga di comando deve essere installato nel `Dockerfile` durante la fase di build del container.
2. `interval`: L'intervallo, in secondi, tra ogni esecuzione del test.
3. `timeout`: Il tempo, in secondi, da attendere prima di ottenere una risposta dal test.
4. `retries`: Quante volte è necessario ripetere lo stesso test se si verifica un errore.

Nel nostro container `app`, il blocco `depends_on` specifica quando questo container deve essere avviato, ovvero quando il container `db` raggiunge lo stato `Healthy`.

Quindi, se eseguiamo `docker compose up -d`, l'intero processo sarà il seguente:

1. `db` si avvia e rimane nello stato `Unhealthy` finché un test non ha successo.
2. `app` si avvia e rimane nello stato `Created` perché `db` è ancora `Unhealthy`.
3. Un test ha successo e `db` diventa `Healthy`.
4. Poiché ora `db` è `Healthy`, `app` diventa `Started` e viene effettivamente avviato.

Conclusione

La procedura descritta sopra rappresenta un flusso normale nel processo di avvio dei container in un contesto Docker Compose. In caso di errori, il modo più rapido per risolvere i problemi di avvio è consultare i log del container con il comando `docker container logs container-id` per verificare perché il container restituisce uno stato di `Error`.