GABRIELE ROMANATO

Come usare PostgreSQL in Docker con Docker Compose e Node.js

Docker semplifica la gestione dei database come PostgreSQL, rendendo facile configurare un ambiente isolato e riproducibile. In questo articolo, vedremo come configurare PostgreSQL in Docker utilizzando Docker Compose e integrarlo con un'applicazione Node.js.

Prerequisiti

- Docker e Docker Compose installati sulla tua macchina.
- Node.js installato (consigliata la versione LTS).

Passo 1: Configurare Docker Compose per PostgreSQL

Inizia creando un file `docker-compose.yml` per configurare un servizio PostgreSQL.

1. Crea una directory per il progetto:

```
mkdir node-postgres-docker
cd node-postgres-docker
```

2. Crea un file `docker-compose.yml` nella directory e aggiungi il seguente contenuto:

```
version: '3.8'

services:
  postgres:
    image: postgres:15
    container_name: postgres_container
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
      POSTGRES_DB: mydatabase
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:
```

3. Avvia PostgreSQL:

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

4. Verifica che il container sia attivo:

```
docker ps
```

Passo 2: Configurare Node.js

1. Inizia creando un'applicazione Node.js:

```
mkdir app
cd app
npm init -y
```

2. Installa le dipendenze necessarie:

```
npm install pg dotenv
```

3. Crea un file `.env` per salvare le variabili di configurazione:

```
DB_HOST=localhost
DB_PORT=5432
DB_USER=user
DB_PASSWORD=password
DB_NAME=mydatabase
```

4. Crea un file `index.js` nella directory `app` con il seguente contenuto:

```
const { Pool } = require('pg');
require('dotenv').config();

const pool = new Pool({
  host: process.env.DB_HOST,
  port: process.env.DB_PORT,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME,
});

async function testConnection() {
  try {
    const client = await pool.connect();
    console.log('Connesso al database!');
    const result = await client.query('SELECT NOW() AS current_time');
    console.log(result.rows[0]);
    client.release();
  } catch (err) {
    console.error('Errore di connessione:', err);
  } finally {
    pool.end();
  }
}
```

```
testConnection();
```

5. Esegui l'applicazione per testare la connessione:

```
node index.js
```

Passo 3: Persistenza dei Dati

Nel file `docker-compose.yml`, abbiamo configurato un volume `postgres_data` per garantire che i dati persistano anche se il container viene eliminato. Puoi verificare che i dati siano memorizzati in:

```
/var/lib/docker/volumes
```

Passo 4: Interazione con il Database

Puoi creare tabelle, eseguire query e interagire con il database tramite il modulo `pg`. Esempio:

1. Modifica `index.js` per creare una tabella:

```
async function createTable() {
  try {
    const client = await pool.connect();
    await client.query(`
      CREATE TABLE IF NOT EXISTS users (
        id SERIAL PRIMARY KEY,
        name VARCHAR(100),
        email VARCHAR(100)
      );
    `);
    console.log('Tabella creata con successo!');
    client.release();
  } catch (err) {
    console.error('Errore durante la creazione della tabella:', err);
  }
}

createTable();
```

2. Esegui di nuovo l'applicazione:

```
node index.js
```

Passo 5: Debug e Risoluzione dei Problemi

- **Errore di connessione al database:** Verifica che il container PostgreSQL sia in esecuzione e che le credenziali siano corrette.

- **Problemi di rete:** Assicurati che la porta 5432 non sia occupata da un altro processo.
- **Variabili d'ambiente:** Controlla che il file `.env` sia configurato correttamente e caricato dall'app Node.js.

Conclusione

Utilizzare Docker Compose per PostgreSQL rende semplice configurare e gestire un database per le applicazioni Node.js. Con pochi comandi, puoi avere un ambiente pronto per lo sviluppo o la produzione. Seguendo questi passaggi, hai tutto ciò che ti serve per iniziare a sviluppare applicazioni scalabili e ben strutturate con Node.js e PostgreSQL.

Applicazioni Correlate

- 

  ### Node.js Placeholder Image

  Applicazione per la generazione con Node.js di immagini segnaposto.
  DockerDocker ComposeNode.jsJavaScriptExpressJS

Node.js URL Shortener

Implementazione in Node.js di un sistema per l'abbreviazione degli URL.
DockerDocker ComposeNode.jsJavaScriptExpressJSMongoDB

- ### JavaScript App Hash Change

  Applicazione che sfrutta gli hash degli URL per gestire contenuto dinamico in JavaScript. DockerDocker ComposeNode.jsJavaScriptExpressJSMySQL