PostgreSQL Docker & Windows Server 2019

Kathegorie
Docker, PostGreSQL, SQL Server

Denise
Robinson

Einführung

PostgreSQL, auch Postgres genannt, ist ein objektrelationales Open Source Datenbankmanagementsystem. Mit MySQL gehört PostgreSQL zu den beliebtesten relationalen Datenbankmanagementsystemen.
Heute ist Postgres eines der häufigsten verwendeten Docker-Images, das in Form von Containern ausgeführt wird. Die Beliebtheit von containerisierten Datenbanken geht auf die Einfachheit zurück, mit der sie bereitgestellt werden. Anstatt eine zentrale Datenbank zu haben, können Entwickler außerdem für jede Anwendung einen PostgreSQL-Container verwenden.

Wir werden uns hier damit beschäftigen, wie man PostgreSQL auf einem Docker Container ausführen und wie man eine PostgreSQL-Datenbank sichern kann.

Voraussetzungen

- Zugriff auf Windows Powershell
- Ein Benutzerkonto mit **Administrator**-Berechtigung
- Eine vorhandene Docker-Installation

PostgreSQL auf Docker Containern ausführen

Das Bereitstellen eines Postgres-Containers ist einfach. Das Postgres-Image zum Erstellen dieser Datenbankcontainer findest Du im offiziellen Docker Hub. Wir zeigen Dir zwei Möglichkeiten, dies zu tun.

Die erste Option verwendet **Docker Compose**, ein Tool zum Verwalten von Docker-Anwendungen mit mehreren Containern. Du kannst Docker Compose verwenden, um Postgres als Dienst, der in einem Container ausgeführt wird, zu konfigurieren. In diesem Fall erstellst Du eine yaml-Datei mit allen Spezifikationen.
Alternativ kannst Du als zweite Option einen **einzelnen Docker-Befehl** mit allen erforderlichen Informationen zum Bereitstellen eines neuen PostgreSQL-Containers verwenden.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Option 1: Postgres mit Docker Compose ausführen

Um einen Postgres-Container mit Docker bereitzustellen, solltest Du diesen auf Deinem System installiert haben.

1. Um eine einfache und saubere Installation zu gewährleisten, wollen wir zuerst ein Arbeitsverzeichnis mit dem Namen **postgres** erstellen und in dieses Verzeichnis wechseln. Das machst Du mit diesem Befehl:

```
mkdir postgres
cd postgres/
```

2. Als Nächstes verwendest Du Docker Compose, um das Postgres-Image herunterzuladen und den Dienst zum Laufen zu bringen. Hierzu erstellst Du mit einem **Editor** Deiner Wahl eine neue **docker-compose.yml**-Datei mit folgendem Inhalt:

```
version: '3.8'
services:
my_service:
image: postgres:latest
ports:
- 5432:5432
environment:
- POSTGRES_PASSWORD=Start123
```

   Die yaml-Konfiguartionsdatei beschreibt, dass es einen **my_service**-Dienst gibt, der auf **latest postgres image** basiert. Es bleibt Dir überlassen, welche Version Du verwenden möchtest. Wir haben uns hier für die neueste '**3.8**' Postgres-Version entschieden. Schließlich musst Du die Ports definieren, über die der Container kommuniziert. **5432** ist die Standardportnummer für PostgreSQL und die Umgebungsvariable für das Passwort des Postgres-Superuser setzen (der Wert "Start123" kann beliebig angegeben werden).

3. Speichere die Datei.

4. Nachdem Du nun die yaml-Konfigurationsdatei hast, kannst Du den Postgres-Dienst starten und den Container ausführen. Verwende den **docker-compose up**-Befehl mit der **-d**-Option um ihn in den Detach-Modus zu versetzen (damit Du weiterhin Befehle von der aktuellen Shell ausführen können):

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

5. Du kannst die dabei ausgegebenen Protokolle mit dem folgenden Befehl überprüfen:

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

Mit **CRTL+C** kehrst Du wieder zur Shell zurück.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Option 2: Postgres mit einem einzelnen Docker-Befehl ausführen

Eine weitere Möglichkeit, PostgreSQL in einem Container bereitzustellen, besteht darin, einen einzelnen Docker-Befehl auszuführen.

Du kannst einen Postgres-Container herunterladen und ausführen, indem Du alle erforderlichen Informationen in einem Befehl angibst.
Folgender Befehl weist Docker an, einen neuen Container unter einem bestimmten Containernamen auszuführen, definiert das Postgres-Passwort und lädt die neueste Postgres-Version herunter:

```
docker run --name [container_name] -e POSTGRES_PASSWORD=[your_
password] -d postgres
```

Bestätige, dass Dein PostgreSQL-Container jetzt aktiv ist, indem Du Docker mit dem unten stehenden Befehl aufforderst, alle laufenden Container aufzulisten:

```
docker ps
```

In unserem Beispiel haben wir einen Container "**postgres**" erstellt, den wir leicht unter anderen laufenden Containern finden können.

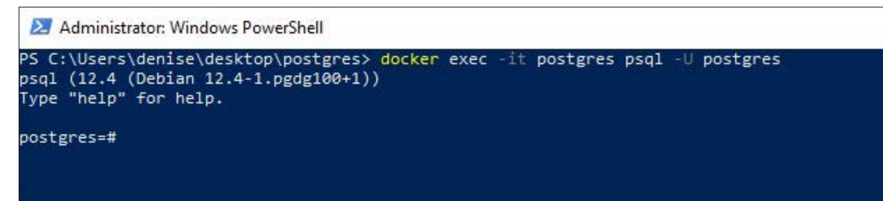
Der Start mit Postgres Containern

Eine Verbindung zu Postgres im Docker Container herstellen

Um in einen Postgres-Container hineinzukommen, muss der folgende Befehl ausgeführt werden:

```
docker exec -it [container_name] psql -U [postgres_user]
```

In diesem Beispiel haben wir uns mit dem Container "**postgres**" als "**postgres**"-Benutzer verbunden.



Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Eine Datenbank erstellen

In unserem Docker Postgres Container können wir mit dem folgenden Befehl eine Datenbank erstellen:

```
create database [db_name];
```

Tipp: Mit **\1** kannst Du Dir alle Datenbanken anzeigen lassen, die auf PostgreSQL ausgeführt werden.

Mit folgendem Befehl verbindest Du Dich als Postgres-Benutzer mit der Datenbank:

```
\c [db_name]
```

Wenn die Datenbank eingerichtet ist, musst Du im nächsten Schritt ein Schema erstellen, mit dem Du eine logische Darstellung der Datenbankstruktur erhältst:

```
create schema [db_schema_name]
```

Hiermit kannst Du eine Tabelle erstellen und Daten zur Tabelle hinzufügen:

```
create table [table_name] ([field_names] [values])
```

Tipp: Um den Postgres-Container zu verlassen benutze **\q**

So sicherst Du eine PostgreSQL-Datenbank mit Docker

Sichern einer lokalen oder einer remote PostgreSQL-Datenbank

1. Befehl zum Sichern einer lokalen oder einer remote PostgreSQL-Datenbank:
```
$ docker run -i postgres /usr/bin/pg_dump
-h [POSTGRESQL_HOST]
-U [POSTGRESQL_USER] [POSTGRESQL_DATABASE] > backup.sql
```

2. Befehl zum Sichern mehrerer PostgreSQL-Datenbanken:
```
$ docker run -i postgres /usr/bin/pg_dumpall
-h [POSTGRESQL_HOST]
-U [POSTGRESQL_USER] > backup.sql
```

3. Befehl zum Sichern einer lokalen oder einer remote PostgreSQL-Datenbank mit Komprimierung (mit gzip):
```
$ docker run -i postgres /usr/bin/pg_dump
-h [POSTGRESQL_HOST]
-U [POSTGRESQL_USER] [POSTGRESQL_DATABASE] | gzip -9 > backup.sql.
gz
```

4. Voriger Befehl, der jedoch das PostgreSQL-Passwort als Umgebungsvariable liefert:
```
$ docker run -i -e PGPASSWORD=[POSTGRESQL_PASSWORD] postgres /usr/
bin/pg_dump
-h [POSTGRESQL_HOST]
-U [POSTGRESQL_USER] [POSTGRESQL_DATABASE] | gzip -9 > backup.sql.
gz
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Sichern einer containerisierten PostgreSQL-Datenbank

1. Befehl zum Sichern einer containerisierten PostgreSQL-Datenbank, wobei eine komprimierte Datei erstellt wird (mit gzip):

```
$ docker exec [POSTGRESQL_CONTAINER] /usr/bin/pg_dump
-U [POSTGRESQL_USER] [POSTGRESQL_DATABASE] | gzip -9 > backup.sql.
gz
```

2. Voriger Befehl, der jedoch zusätzlich das PostgreSQL-Passwort als Umgebungsvariable bei vorhandenem Container festlegt:

```
$ docker exec [POSTGRESQL_CONTAINER] /bin/bash
-c "export PGPASSWORD=[POSTGRESQL_PASSWORD]
&& /usr/bin/pg_dump -U [POSTGRESQL_USER] [POSTGRESQL_DATABASE]"
| gzip -9 > backup.sql.gz
```

Du hast nun zwei verschiedene Methoden zum Ausführen von PostgreSQL in einem Docker-Container kennengelernt und weißt wie man verschiedene Arten von Datenbanken sichern kann. Damit solltest Du jetzt mit der Erstellung Deiner Datenbanken mit allen erforderlichen Daten beginnen können

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

backup, database, docker, mssql,
powershell, sql server, t-sql, volume

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612