Erstellen eines MSSQL Docker Container

Container Database-Entwicklung auf höchstem Niveau

Leicht, schnell und praktisch

Du brauchst keinen Administrator oder umfangreiche Anleitungen über den MS SQL Server. Konzentriere Dich hier auf das, was Du wirklich benötigst:

- Einen SQL Server, der Anfragen annehmen und Daten verarbeiten kann.
- Deine API (Programmierschnittstelle) kann in Deiner Datenbank getestet werden.

In den ersten Stufen der Entwicklung solltest Du nicht viel Zeit mit der Konfiguration Deines SQL Servers oder der Konfiguration einer gehosteten Umgebung investieren. Hier kommt der lokale Docker Container ins Spiel, der Deine Arbeit positiv verändern wird. Dieser Artikel erklärt den Aufbau eines Docker Containers auf der Basis von mcr.microsoft.com/mssql/server, um ein benutzerdefiniertes Dockerfile zu erstellen, das einen SQL Server starten kann, eine Datenbank und ein Table erstellen kann sowie das Table mit Daten befüllen kann. Der Artikel zeigt Dir außerdem, wie Du eine .NET Core Programmierschnittstelle erstellst. Die fertigen MSSQL Dockerfile und Skripte kannst Du über unseren GitHub Account downloaden.
Ein Dank gilt hier twright-msft für seine stellar docker images, die ich sehr oft nutze.

Image-Test

Bevor wir eine benutzerdefinierte Docker-Datei erstellen, muss das Image heruntergeladen sowie geöffnet werden, damit sichergestellt werden kann, dass es sich öffnen lässt. Außerdem muss Docker auf Deinem Computer installiert sein.

Starte mit folgender Befehlszeile:

```
docker pull mcr.microsoft.com/mssql/server:latest
```

Öffne das Image und starte den Container mit dem folgenden Code.

```
docker run \
- e 'ACCEPT_EULA=Y' \
- e 'SA_PASSWORD=Password1!' \
- e 'MSSQL_PID=Express' \
--name sqlserver \
- p 1433:1433 -d mcr.microsoft.com/mssql/server:latest
```

Kathegorie
SQL Server

Benedikt
Schackenberg

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

Der SQL Server Container sollte mit dem Port 1433 laufen, wenn Du folgenden Code startest:

```
docker container ls
```

Öffnen des Containers

Starte die folgende Befehlszeile um sich mit dem Container zu verbinden:

```
docker exec -it sqlserver "bash"
```

Starte sqlcmd in der Kommandozeile des Containers. Hier wird das gleiche Passwort benötigt, das wir für das Erstellen des Containers mit docker run benutzt haben.

```
/opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P "Password1!"
```

Wenn Du jetzt eine 1> siehst, hast Du bis hier hin alles richtig gemacht. Dieses Zeichen zeigt Dir, dass Du Dich in der richtigen sqlcmd Befehlszeile befindest und mit Deinem ausgewählten Server arbeiten kannst.

Wir starten nun mit SQL um eine Datenbank und ein Table zu erstellen. Nutze dafür folgenden Code:

```
CREATE DATABASE testdb
GO
USE testdb
CREATE TABLE Testtab (id INT, name VARCHAR(50))
INSERT into Testtab VALUES (1, "Wonder Woman")
GO
```

Überprüfe nun die Datenbank mit folgendem Code, um sicherzustellen, dass alles korrekt geschrieben wurde:

```
SELECT * FROM Testtab;
GO
```

Fertig!

Erstellen des Dockerfiles

Die händische Eingabe der Kommandos kann etwas umständlich sein. Daher wollen wir alle Befehle in Skripte hinterlegen.

1. **entrypoint.sh** – Das Skript wurde beim Start ausgeführt und führt einfach import-data.sh aus und startet den SQL Server.
2. **import-data.sh** – führt ein sqlcmd aus, das setup.sql aufruft, und einen bcp-Befehl, der eine CSV-Datei importiert.
3. **setup.sql** – SQL-Skript, das eine Datenbank und eine Tabelle erstellt

Das folgende Dockerfile beinhaltet alle drei Skripte - wir erklären diese später im genauen Detail, damit keine weiteren Fragen aufkommen.

Dockerfile

```
FROM microsoft/mssql-server-linux:latest# Create work directory
RUN mkdir -p /usr/work
WORKDIR /usr/work# Copy all scripts into working directory
COPY . /usr/work/# Grant permissions for the import-data script to
be executable
RUN chmod +x /usr/work/import-data.shEXPOSE 1433CMD /bin/bash ./
entrypoint.sh
```

Starte Deine Datenbank Einstellung der MSSQL docker docs vor dem finalen Befehl, den SQL Server zu starten. Mit der Datei entrypoint.sh wird die Datei import-data.sh vor dem Öffnen des Servers gestsrtet.

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

entrypoint.sh

```
/usr/work/import-data.sh & /opt/mssql/bin/sqlservr
```

Bei der Container-Erstellung öffnet entrypoint die Datei import-data.sh und startet den Server. Diese Datei stellt uns zwei SQL Server Befehle zur Verfügung sowie den notwendigen Befehl, die Datenbank ohne Unterbrechung für 90 Sekunden zu starten. (Nutze die Zeit um Dir einen Kaffee zu machen oder kurz zu meditieren – der Computer sollte in dieser Zeit nicht benutzt werden)

import-data.sh

```
# wait for the SQL Server to come up
sleep 90s#run the setup script to create the DB and the schema in
the DB
/opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P "Password1!" -i
setup.sql# import the data from the csv file
/opt/mssql-tools/bin/bcp testdb.dbo.Testtab in "/usr/work/testdb.
csv" -c -t',' -S localhost -U SA -P "Password1!" -d testdb
```

Der zweite Befehl sqlcmd öffnet die Datei setup.sql, die das Grundgerüst unserer Datenbank bildet. Sobald die Datenbank erstellt wurde, importiert sie mit Hilfe von bcp Daten aus einer .csv-Datei.

setup.sql

```
CREATE DATABASE testdb;GOUSE testdb;GO
CREATE TABLE Testtab (id INT, name VARCHAR(50));GO
```

Baue das Image mit Deinem Dockerfile

Markiere Dein Image mit folgendem Code:

```
docker build -t mssql:dev .
```

Öffne das Image mit docker run nach der Fertigstellung des vorigen Codes. Dieser Vorgang ist fast identisch dem vorigen docker run, mit dem Unterschied, dass wir das neue mit mssql:dev markiert haben.

```
docker run \
- e 'ACCEPT_EULA=Y' \
- e 'SA_PASSWORD=Password1!' \
- e 'MSSQL_PID=Express' \
--name sqlserver \
- p 1433:1433 \
- d mssql:dev
```

Sobald das Image fertiggestellt wurde und die Image ID bestätigt wurde, öffne den Container um zu bestätigen, dass es funktioniert:

```
docker exec -it sqlserver "bash"
```

Aus dieser Container Befehlszeile kannst Du die testdb database öffnen:

```
/opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P "Password1!" -d
testdb
```

Prüfe die Daten auf ihre Richtigkeit, bevor Du fortfährst. Es kann vorkommen, dass Daten fehlen, deswegen ist dieser Schritt sehr wichtig.

```
SELECT * FROM Testtab;
GO
```


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

Das wars schon! Jetzt solltest Du einen SQL Server Datenbank erstellt haben, die Du einfach konfigurieren und nach Deinen Wünschen veröffentlichen kannst.

Lösche Deinen Container mit folgender Befehlszeile:

```
docker kill sqlserver
```

Starte Deinen Container neu mit dem Befehl docker run in einem Skript, etwa mit der folgenden Datei start-docker-sql.sh oder einer ähnlichen Datei

```
docker run \
- e 'ACCEPT_EULA=Y' \
- e 'SA_PASSWORD=Password1!' \
- e 'MSSQL_PID=Express' \
--name sqlserver \
- p 1433:1433 \
- d mssql:dev
```

Veröffentlichung auf Docker Hub

Dein fertiges working image solltest Du unbedingt auf einem Host speichern, damit Du sie von dem Docker Hub in verschiedene Kanäle streuen kannst.
Erstelle Dir dafür einen Docker Hub-Account auf hub.docker.com.
Erstelle dann ein neues Profil (repository) mit einem Namen Deiner Wahl und einer kurzen Beschreibung.
Gib nun folgendes in Deiner Befehlszeile ein, damit Dir die Zugangsdaten für Deinen Account angezeigt werden:

```
docker login
```

Markiere nun Dein Bild mit folgendem Code:

```
docker tag local-image:tagname username/new-repo:tagname
```

Dupliziere Dein Image nun auf einen neuen Speicherort mit folgendem Code:

```
docker push username/new-repo:tagname
```

Abschließend kannst Du Dein Image mit folgendem Befehlskommando pullen:

```
docker pull username/new-repo:tagname
```

Dieser Vorgang sollte Deine Arbeit um einiges erleichtern!

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