Installatie handleiding

Onderbeke Niels

Table of Contents

[Intro 3](#_Toc94109917)

[Benodigdheden 3](#_Toc94109918)

[Software 3](#_Toc94109919)

[Bestanden 3](#_Toc94109920)

[Kennis 3](#_Toc94109921)

[Environment variabelen 4](#_Toc94109922)

[Installatie apollo-client, apollo-client-persisted-queries & redis-server-cache branch 6](#_Toc94109923)

[Info 6](#_Toc94109924)

[Volledige Docker omgeving 6](#_Toc94109925)

[Niet volledige Docker omgeving 7](#_Toc94109926)

[Installatie graphCDN banch 8](#_Toc94109927)

[Info 8](#_Toc94109928)

## Intro

In deze installatie handleiding zal u te weten komen hoe u mijn researchproject kan gaan installeren. Meer bepaald hoe u de Github branches van de best gevonden caching strategiën kan pullen en kan gaan runnen op uw eigen device.

## Benodigdheden

### Software

Belangrijk om te weten is dat de database server & Redis server default in een Docker container draaien. Er is een docker-compose file voorzien in de root folder van mijn Github repository.

U kan het volledige project in een Docker omgeving op gaan zetten. De apollo-client, apollo-client-persisted-queries, redis-server-cache & graphCDN branches op mijn Github repository zijn voorzien van een docker-compose file met daarin alle nodige services.

Onafhankelijk van volledige Docker omgeving:

* [Google Chrome browser](https://www.google.com/chrome/?brand=YTUH&gclid=Cj0KCQiA_8OPBhDtARIsAKQu0gaAiLOBjVgKawT4UvcTXtzUD_xc7eZ2QV_cFqG1vXO4c7mZa5wu2YMaAgmwEALw_wcB&gclsrc=aw.ds) (Dev tools caching clients)
* [Docker](https://www.docker.com/products/docker-desktop)
* [Git](https://git-scm.com/)
* [Azure Data Studio](https://docs.microsoft.com/en-us/sql/azure-data-studio/download-azure-data-studio?view=sql-server-ver15) of [SQL Server Management Studio](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15)
* [Apollo Client Dev Tools](https://chrome.google.com/webstore/detail/apollo-client-devtools/jdkknkkbebbapilgoeccciglkfbmbnfm)

Zonder volledige Docker omgeving:

* [Node.js v16.13.2](https://nodejs.org/en/)
* [SQL Server Management Studio](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15)
* [Redis](https://redis.io/)

### Bestanden

Om dit project te kunnen gaan testen heeft u de **10 GB Stack Overflow dump** database nodig. Deze kan u vinden via deze [link](https://www.brentozar.com/archive/2015/10/how-to-download-the-stack-overflow-database-via-bittorrent/). Afhankelijk van welk device je gebruikt kan deze database traag zijn in het queriën van relaties. Om dit te versnellen kan je op de foreign keys een indexen gaan plaatsen.

De code & nodige Dockerfiles kan u vinden op mijn Github repository

( <https://github.com/OnderbekeNiels/research-project-3mct/tree/main> ).

De verschillende caching strategieën zijn gegroepeerd per **branch**.

### Kennis

Om dit project te gaan installeren heeft u een goede basiskennis nodig van verschillende development technologieën zoals Node.js, Docker & SQL databases. Om dit project te kunnen testen en begrijpen is er enige kennis over caching, Redis en React.js vereist.

U kan meer informatie over dit project verkrijgen via deze Medium blog post.

<https://medium.com/@niels.onderbeke.no/research-project-which-is-the-best-caching-strategy-with-graphql-for-a-big-relational-database-56fedb773b97>

### Environment variabelen

Database

**.env**

MSSQL\_SA\_PASSWORD=Strong(!)Password

ACCEPT\_EULA=Y

Client-side

**.env**

Deze geldt voor alle branches, uitgezonderd de GraphCDN branch.

NEXT\_PUBLIC\_BACKEND\_URL=http://localhost:4000/graphql

GraphCDN lokale ontwikkel omgeving

NEXT\_PUBLIC\_BACKEND\_URL=http://localhost:3010

Server-side

**ormconfig.json**

{

"type": "mssql",

"host": "localhost",

"port": 1433,

"username": "sa",

"password": "Strong(!)Password",

"database": "StackOverflow",

"entities": ["src/entity/\*\*/\*.ts"],

"synchronize": false,

"extra": { "trustServerCertificate": true, "encrypted": false },

"requestTimeout": 300000

}

**docker.ormconfig.env**

TYPEORM\_CONNECTION=mssql

TYPEORM\_HOST=mssqldb

TYPEORM\_PORT=1433

TYPEORM\_USERNAME=sa

TYPEORM\_PASSWORD=Strong(!)Password

TYPEORM\_DATABASE=StackOverflow

TYPEORM\_SYNCHRONIZE=false

TYPEORM\_LOGGING=false

TYPEORM\_ENTITIES=dist/entity/\*\*/\*.js

TYPEORM\_DRIVER\_EXTRA='{"trustServerCertificate": true, "encrypted": false}'

## Installatie apollo-client, apollo-client-persisted-queries & redis-server-cache branch

### Info

Github branches

<https://github.com/OnderbekeNiels/research-project-3mct/tree/apollo-client>

<https://github.com/OnderbekeNiels/research-project-3mct/tree/apollo-client-persisted-queries>

Belangrijk: Om Apollo Client Dev Tools in Google Chrome te kunnen gebruiken zal je de development client-server moeten starten. De Apollo Client Dev Tools zijn niet beschikbaar in een productie build. De client-server-next Docker container is ook een productie build.

### Volledige Docker omgeving

1. Clone de Github repository en schakel over naar de gewenste branch.
2. Controleer of de nodige .env files aanwezig zijn in het project. Is dit niet het geval, voeg deze dan toe met de variabelen die u hierboven kan vinden.
3. Download de 10 GB Stack Overflow database.
4. Unzip de gedownloade map en plaats de files in een map met de naam **stackoverflowdb** of hernoem de map naar **stackoverflowdb.**
5. Plaats de stackoverflowdb map op hetzelfde niveau als de repository map.

Graphical user interface, application

Description automatically generated with medium confidence

1. Controleer of Docker runt op jouw pc. Zo niet, start het op.
2. Zet alle services in de **docker-compose.yml** file in commentaar uitgezonderd de mssqldb service.
3. Voer volgend commando uit in de terminal => **docker-compose -f "docker-compose.yml" up -d --build**
4. Open Azure Data Studio of SSMS en verbind met de server via deze credentials:

Graphical user interface, text, application, email

Description automatically generated

1. Voor deze query uit op de server om de mdf file te importeren als database.

CREATE DATABASE StackOverflow

ON

(name = so1, filename = N'/var/opt/mssql/data/StackOverflow2010.mdf')

FOR ATTACH;

1. Haal alle services terug uit commentaar in de docker-compose file en voer volgend commando uit => **docker-compose -f "docker-compose.yml" down** & vervolgens => **docker-compose -f "docker-compose.yml" up -d --build**
2. De client-side applicatie is beschikbaar via volgend adres => **http://localhost:3000/**

### Niet volledige Docker omgeving

Om de database te runnen volgt u de stappen zoals in de Docker sectie maar hoeft u de andere services niet meer uit commentaar te halen.

Om de Next.js & Express server op jouw eigen Node omgeving te runnen moet je volgende commando’s uitvoeren in de client-app & typegql-sever mappen: npm install & npm run dev.

Op deze manier start je de development build op van beide applicaties.

## Installatie graphCDN branch

### Info

Github branch

<https://github.com/OnderbekeNiels/research-project-3mct/tree/graphCDN>

Om te starten moet u exact dezelfde stappen volgen zoals bij **Installatie apollo-client & apollo-client-persisted-queries banches.** De basis is hier hetzelfde, alleen wordt er gebruik gemaakt van een andere branch met een andere fetch URL in de client-app.

Het verschil met de andere installaties is de toevoeging van GraphCDN. Je hebt namelijk eerst een account en service nodig bij GraphCDN zelf. Maak via deze [link](https://graphcdn.io/signup) een account aan en geef je service een naam. Er zal je gevraagd worden jouw graphQL server adres mee te geven, voorlopig heb je deze nog niet dus negeer dit even.

Nu heb je 2 opties, je kan gebruik maken van de develop omgeving van graphCDN met de develop CLI (aangeraden) of je tunnelt jouw lokale GraphQL server IP naar een domein naam via een tool zoals ngrok. In deze handleiding wordt enkel de eerste optie behandeld.

Om de CLI te gebruiken, volg je de stappen zoals hier beschreven: <https://docs.graphcdn.io/docs/how-to-use-graphcdn-in-development> . Om de effectieve caching te gaan testen kan je zelf caching rules definiëren op het GraphCDN dashboard en deze vervolgens pullen met **graphcdn pull.** Of je gebruikt de rules die in de **graphcdn.yml** file staan in de Github repository map**.**

Wanneer dit allemaal inorde is voer je enkel nog het commando **graphcdn serve --backend-port 4000 --service stackof-rp --path /graphql** uit.

Nu zal er op poort 3010 een proxy draaien die zich zal gedragen als de CDN. Deze zal vervolgens aangesproken worden door de client-app.