Het ip adres van de docker host is te vinden via de docker quickstart terminal in "C:\Program Files\Docker Toolbox"

(start.sh)

**(Dit tool bestaat overigens niet op de MAC. Kennelijk is het ook nog verbonden met de VirtualBox van Oracle). Gebruik liever docker container ls om achter het adres van de docker host te komen).**

Opstarten in deze terminal duurt best een poos, de default docker vm wordt hier opgestart.

Omdat we met docker toolbox werken (geen pro versie van windows 10) moet een ander ip adres gebruikt worden ipv localhost

Vanuit deze terminal geeft het volgende commando het ip adres:

docker-machine ip

De laatste keer was dit (en dat zal wel zo blijven)

192.168.99.100

**Echter op de MAC is het adres 0.0.0.0. Zie verderop over de volgorde van docker run.**

De docker voorbeelden uit bijvoorbeeld https://docs.docker.com/get-started/part2/ werken via dit ip adres,

zo zal het python voorbeeld dat wordt opgestart via docker run -p 4000:80 friendlyhello

te vinden zijn op het adres vanuit de browser:

http://192.168.99.100:4000/

start postgres docker image op in een container met

docker run --name mypostgres --net host -e POSTGRES\_PASSWORD=mysecretpassword -d -p 5432:5432 postgres

**NB de volgorde maakt uit. Op de MAC werkt dit niet: de poorten worden niet ge-exposed. Liever:**

**docker run --name mypostgres -d -p 5432:5432 -e POSTGRES\_PASSWORD=mysecretpassword postgres**

Via het commando

docker container ls

is het mogelijk om te achterhalen wat het id is van dit docker proces laten we zeggen

309a6a850721

Start nu van binnen deze container de postgres sql op:

docker exec -it 309a6a850721 bash

Vanuit de bash prompt de postgres sql starten als gebruiker postgres

psql -U postgres

Dan connecten met de juiste database

\c mytest

Hier kun je de tabel infotable bekijken:

select \* from infotable;

Database aanmaken

create database mytest;

Table aanmaken

create table infotable(naam varchar(30));

insert into infotable(naam) values ('adriaan');

commit hoeft niet, autocommit.

Vanuit tooltje TablePlus een connectie maken, name een goede naam (postgres default db) Tag (development)

host het machine ip adres van docker - in plaats van localhost (192.168.99.100), port 5432, user postgres en

password mysecretpassword.

het is bijna 19:00 ik ga eten!

Als xxxx het korte id is van de container (te zien bij docker container ls):

docker container stop xxxx

kan de container weer opnieuw worden opgestart met behoud van data

probeer eerst

docker container start xxxx

Dit geeft een foutmelding

docker: Error response from daemon: Conflict. The container name "/mypostgres01" is already in use by

container "309a6a850721161447165d9ec0f0da3e2b9f0d1e36d83ff9592a6d9ec57577f9".

Deze lange id (laten we zeggen xxxxxxxxxxxxx) gebruiken om de container opnieuw op te starten:

docker container start xxxxxxxxxxxxx

De data is dan nog steeds aanwezig. Op dezelfde manier kun je een container opnieuw opstarten nadat de pc opnieuw is

opgestart. Ook dan geldt dat de data er dan nog steeds is.

Na een container remove van de postgres container is de data verdwenen. Bedoeling is om hier een volume van

te maken dat bewaard blijft nadat de container is verwijderd.

Ubuntu gebruiken

docker run --name ubuntu\_bash --rm -i -t ubuntu bash

Volume maken

docker volume create my-vol

Volume inspecteren

docker volume ls

Volume verwijderen

docker volume rm my-vol

|  |
| --- |
| ik werd hier helemaal niet lekker van: Ik kwam er niet goed achter hoe “live” een lokale folder kon worden gebruikt om met een docker container bestanden uit te wisselen: wat betekent onderstaande nu allemaal?? |
| Start container met een volume  docker run -d \  --name devtest \  --mount source=myvol2,target=/app \  nginx:latest |
| dus heb ik maar met een voorbeeld gedaan waarin een dockerfile werd gebruikt om een image te maken. Dockerfile: |
| # Use an official Ubuntu runtime as a parent image  FROM ubuntu:latest  # Set the working directory to /app  WORKDIR /app  # Copy the current directory contents into the container at /app  COPY . /app |
| docker image build -t ubuntu:1.0 .  en dan  docker container run --name ubuntu --rm -i -t ubuntu:1.0 bash |