# Docker

## Dockerfile start

React dockerfile example

```
FROM node
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

.dockerignore example for react

```
node_modules
Dockerfile
.git
.gitignore
.dockerignore
.env
```

## commands

Docker file builden

```
docker build -t react-image .
```

Lijst van docker images zien

```
docker image ls
```

Docker image verwijderen

```
docker image rm [id]
```

Docker image runnen

```

```

```
net stop winnat
docker run -d -p 3000:3000 --name react-app react-image
net start winnat
```

-p: voor de poort in te stellen, in dit geval alle traffic van 3000 gaat naar poort 3000 -d: voor in de achtergrond, --name voor makkelijke name

lijst van docker images die aan het runnen zijn zien

```
docker ps
```

docker image stopzetten

```
docker rm react-app -f
```

in de docker container src zelf gaan

```
docker  exec -it react-app bash
```

## bind mount to sync src changes

run met auto update **WATCHPACK_POLLING=true** kan in env file

```
docker run -v dirlocaldirectory:containerdirectory -d -p 3000:3000 --name react-
app react-image
```

example

```
docker run -e WATCHPACK_POLLING=true -v
C:\Users\guusd\OneDrive\Bureaublad\Zelfstudie\Docker\testing-dev-
container\src:/app/src:ro -d -p 3000:3000 --name react-app react-image
```

of

powershell

```
docker run -e WATCHPACK_POLLING=true -v ${pwd}\src:/app/src:ro -d -p 3000:3000 --
name react-app react-image
```

commandline

```
docker run -e WATCHPACK_POLLING=true -v %cd%\src:/app/src:ro -d -p 3000:3000 --
name react-app react-image
```

## environment variables docker

aan de Dockerfile toevoegen

```
ENV REACT_APP_NAM=myname
```

**OF** cmd

```
docker run -e WATCHPACK_POLLING=true -v %cd%\src:/app/src:ro -d --env-file .env -p
3000:3000 --name react-app react-image
```

powershell

```
docker run -e WATCHPACK_POLLING=true -v ${pwd}\src:/app/src:ro -d --env-file .env
-p 3000:3000 --name react-app react-image
```

OF

```
docker run -v %cd%\src:/app/src:ro -d --env-file ./.env -p 3000:3000 --name react-
app react-image
```

## Docker compose

zorgt ervoor dat de lange run command niet meer moet komen

in docker-compose.yml file

```
version: '3'
services:
  react-app:
    build: .
    ports:
      - '3000:3000'
    volumes:
      - ./src:/app/src
    #environment: andere manier om env dingen mee te geven
```

```
    #  - REACT_APP_NAME = "React App"
    #  - WATCHPACK_POLLING=true
    env_file:
      - ./.env
```

runnen

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

alles sluiten

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

runnen met opnieuw build voor dockerfile changes

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

## production & dev environment afstemmen

Development dockerfile veranderen van naam naar **Dockerfile.dev**

docker build veranderd nu naar

```
docker build -f Dockerfile.dev
```

docker compose moet ook veranderd worden naar

```
version: '3'
services:
  react-app:
    build:
      context: .
      dockerfile: Dockerfile.dev
    ports:
      - '3000:3000'
    volumes:
      - ./src:/app/src
    #environment:
    #  - REACT_APP_NAME = "React App"
    #  - WATCHPACK_POLLING=true
    env_file:
      - ./.env
```

### multi stage build (dev en prod)

Dockerfile.prod code

```
FROM node as build
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .
RUN npm run build

FROM ngninx
COPY --from=build /app/build /usr/share/nginx/html
```

prod image builden

```
docker build -f Dockerfile.prod -t docker-image-prod .
```

prod image runnen

```
docker run --env-file ./.env -d -p 8080:80 --name react-app-prod docker-image-prod
```

## compose met dev en prod

algemene compose file

```
version: '3'
services:
  react-app:
    build:
      context: .
      dockerfile: Dockerfile.dev
```

development compose

```
version: '3'
services:
  react-app:
    build:
      context: .
      dockerfile: Dockerfile.dev
    ports:
      - '3000:3000'
```

```
    volumes:
      - ./src:/app/src
    #environment:
    #  - REACT_APP_NAME = "React App"
    #  - WATCHPACK_POLLING=true
    env_file:
      - ./.env
```

prod compose

```
version: '3'
services:
  react-app:
    build:
      context: .
      dockerfile: Dockerfile.prod
    ports:
      - '8080:80'
    env_file:
      - ./.prod.env
```

runnen van dev

--build alleen maar eerste keer nodig daarna niet meer

```
docker-compose -f docker-compose.yml -f docker-compose-dev.yml up -d --build
```

afzetten

```
docker-compose -f docker-compose.yml -f docker-compose-dev.yml down
```

arguments doorgeven in ymls (beter gewoon met aparte .env files werken)

```
version: '3'
services:
  react-app:
    build:
      context: .
      dockerfile: Dockerfile.prod
      args:
        - REACT_APP_NAME=name
    ports:
      - '8080:80'
```

in Dockerfile

```
FROM node as build
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .
ARG REACT_APP_NAME
ENV REACT_APP_NAME=$REACT_APP_NAME
RUN npm run build

FROM nginx
COPY --from=build /app/build /usr/share/nginx/html
```

## only run a few stages

```
docker build --tager build -f Dockerfile.prod -t multi-stage-example .
```