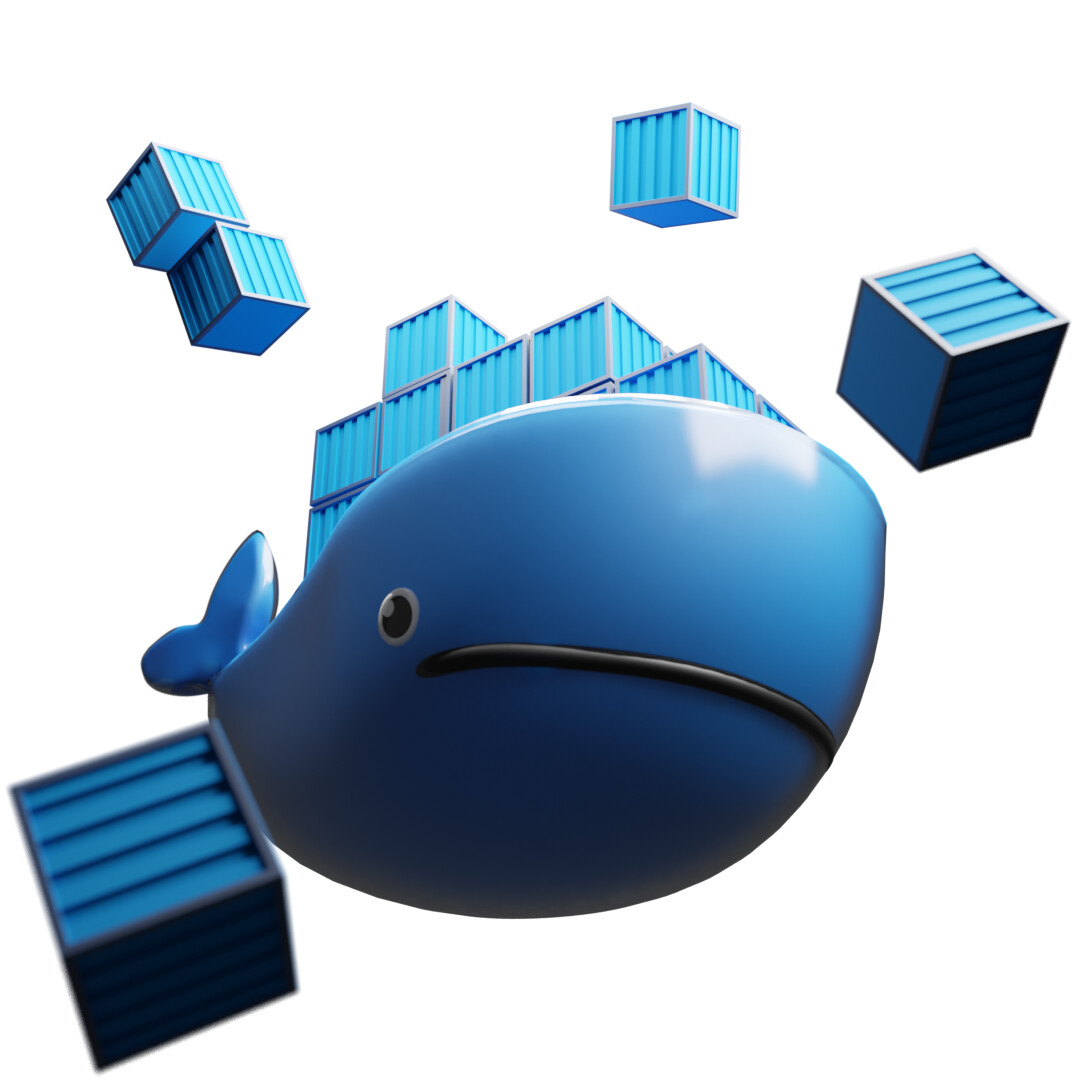
# Hoe implementeer ik Docker om de onderhoudbaarheid en gebruiksvriendelijkheid van mijn applicatie te verbeteren.



Content

[Hoe implementeer ik Docker om de onderhoudbaarheid en gebruiksvriendelijkheid van mijn applicatie te verbeteren. 1](#_Toc151552529)

[Hoe zit Docker in elkaar en hoe kan dit worden gebruikt? 3](#_Toc151552530)

[Hoe kan Docker geïmplementeerd worden in een bestaande multi-layer applicatie. 4](#_Toc151552531)

[Bronnen 8](#_Toc151552532)

# Hoe zit Docker in elkaar en hoe kan dit worden gebruikt?

Docker is een platform dat gebruikt kan worden om applicaties in aparte virtuele containers te runnen, waardoor deze onafhankelijk is van de lokale infrastructuur van de gebruiker. Hierdoor kunnen deze virtuele containers makkelijk gedeeld worden tussen computers zonder dat de applicatie wel bij de een werkt maar niet bij de ander. Door dit te combineren met Dockers functies van testen en deployen, kan de applicatie sneller van coderen naar productie gaan. Een Docker container bevat alle code, de run-time omgeving, dependencies en instellingen in één pakket.

**Geïsoleerde containers**

Docker maakt gebruik van containers, waarin alle dependencies van de applicatie worden gedraaid. Zo zijn alle benodigdheden van de applicatie beperkt tot de container zodat deze niet lokaal geïnstalleerd hoeven te worden. Hierdoor wordt de onderliggende infrastructuur niet verstoort en kunnen er ook geen conflicten met andere applicaties ontstaan. Ook kunnen containers snel en makkelijk gestopt en gestart worden.

Het is ook mogelijk om meerdere containers te draaien voor een applicatie en deze met elkaar te laten communiceren. Dit kan gebruikt worden in een architectuur met meer lagen.

Een webapplicatie bijvoorbeeld kan dus een front-end container, back-end container en een database container hebben.

**CI/CD**

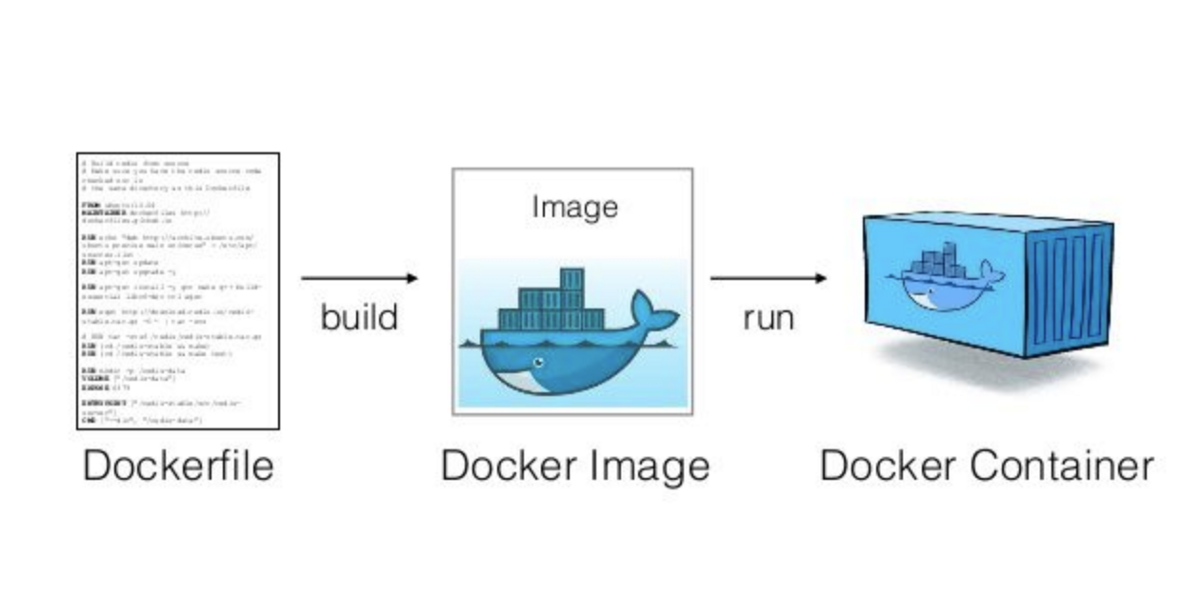
Met docker is het mogelijk om lokaal code te schrijven en deze direct naar een test omgeving te pushen waar de applicatie automatisch wordt getest met continuous integration. Vervolgens kunnen andere gebruikers de geüpdate image direct gebruiken. Docker containers kunnen op verschillende manieren gebruikt worden, zo is het mogelijk om containers lokaal of op een cloud service te draaien. Voor servers is Docker een groot voordeel omdat Docker containers vaak efficiënter zijn dan de applicatie lokaal draaien, waardoor er meer diensten naast elkaar kunnen worden gedraaid met dezelfde hardware.

**Integratie cloudservices**

Met Docker images kan er automatic deployment worden toegepast in combinatie met een Cloud service zoals Azure, AWS en Google Cloud. Hierdoor is een applicatie direct globaal beschikbaar. Containers vereisen minder hardware omdat ze OS-Kernel delen.

# Hoe kan Docker geïmplementeerd worden in een bestaande multi-layer applicatie.

Om gebruik te maken van Docker is het noodzakelijk om een aantal stappen te doorlopen. Zo moet de applicatie eerst van een Docker file worden voorzien. Vervolgens kan hiermee een Docker image worden aangemaakt. Deze image kan vervolgens worden gedraaid in een container:



(Blogger, 2022)

**Docker file**

De Docker file is een file die de configuratie bevat om de applicatie te builden voor de Docker Image. Hier is een voorbeeld van een Dockerfile:

# Start your image with a node base image

FROM node:18-alpine

# The /app directory should act as the main application directory

WORKDIR /app

# Copy the app package and package-lock.json file

COPY package\*.json ./

# Copy local directories to the current local directory of our docker image (/app)

COPY ./src ./src

COPY ./public ./public

# Install node packages, install serve, build the app, and remove dependencies at the end

RUN npm install \

    && npm install -g serve \

    && npm run build \

    && rm -fr node\_modules

EXPOSE 3000

# Start the app using serve command

CMD [ "serve", "-s", "build" ]

Belangrijke instructies in een Dockerfile kunnen zijn:

* ‘From’: om de basisimage te specificeren.
* ‘Copy’ of ‘Add’: om lokale bestanden naar de image te kopiëren.
* ‘Run’: om commando's uit te voeren voor het installeren van dependencies.
* ‘Expose’: om aan te geven op welke poorten de applicatie binnen de container beschikbaar zal zijn. Dit is de interne poort van de container.

Wanneer de Docker image wordt gemaakt, worden de instructies uit de Docker file gebruikt. Zo worden de lokale files gekopiëerd naar de Docker Image, en worden de dependencies geïnstalleerd. Vervolgens wordt aangegeven op welke port de container moet gaan draaien.

Belangrijk hierbij is dat de expose port die hier te zien is de interne port is waarop de applicatie luistert wanneer deze in een container wordt gestart. De externe port waarop de container draait zal apart ingesteld worden bij het starten van de container.

**.dockerignore**

Naast een Dockerfile is het ook belangrijk om een .dockerignore file aan te maken. Hierin worden files en directories gezet die niet in de image mee hoeven te worden genomen. Een voorbeeld hiervan zijn de node\_modules in een React applicatie. Deze worden namelijk aangemaakt wanneer de dependencies worden geïnstalleerd, en dit wordt al gedaan bij het aanmaken van de Docker image.

**Bouwen Docker Image**

Nadat er een Dockerfile en een .dockerignore file is gemaakt, kan er een Docker Image worden aangemaakt. Wanneer Docker desktop is geïnstalleerd kan dit gedaan worden met de volgende command:

docker build -t naam-van-applicatie .

Hierbij geeft de -t parameter aan dat de “tag” (naam) van de image wordt ingevoerd. De punt “.” Op het einde geeft aan waar deze image op wordt gebaseerd, in dit geval de huidige directory waar het command wordt uitgevoerd.

**Docker Container**

Nadat de image is gebouwd, kan deze worden gebruikt om een container te starten. Dit kan gedaan worden via Docker Desktop of via de command line met docker run. Bij het gebruik van docker run, kun je verschillende opties specificeren, zoals:

* -p: om de poortmapping tussen de host en de container in te stellen.
* -d: om de container in de achtergrond te draaien.
* -it: voor een interactieve terminal binnen de container.

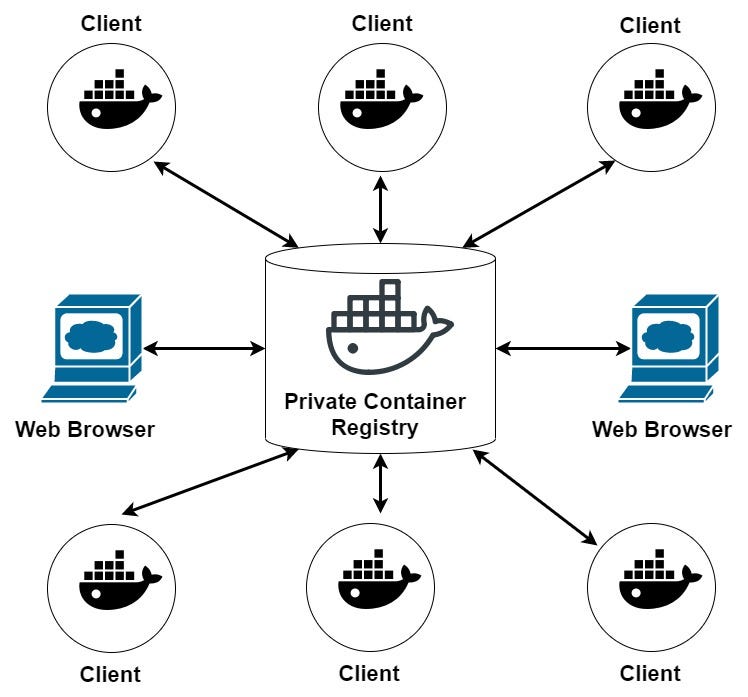
Een voorbeeld van het starten van een container met een poortmapping is:

docker run -p 80:8080 naam-van-applicatie

Hier wordt de interne poort 8080 van de container gemapt op poort 80 van de host.

**Gebruik van Docker Hub en Registries**

Docker images kunnen worden opgeslagen en gedeeld via Docker Hub of andere registry services. Je kunt images rechtstreeks van deze registries halen met docker pull of ze pushen met docker push voor het delen met anderen.



(Blue, 2022)

**Netwerken en Volumes**

In meer geavanceerde Docker-gebruiksscenario's kun je Docker-netwerken en volumes gebruiken voor respectievelijk inter-containercommunicatie en data-persistentie. Netwerken zorgen voor veilige communicatie tussen containers, terwijl volumes het mogelijk maken om gegevens op te slaan buiten de levensduur van individuele containers.



(*Use volumes*, 2020)

# Bronnen

*“Docker Overview”*. (2023, 16 november). Docker Documentation. <https://docs.docker.com/get-started/overview/>

Acharya, V. (2023, 19 juli). How to deploy Docker Containers on AWS - Vishwas Acharya - Medium. *Medium*. https://medium.com/@vishwasacharya/mastering-docker-container-deployment-on-aws-expert-tips-tricks-5b73b36296aa

Blogger, G. (2022, 24 november). *Differences between a dockerfile, docker image and docker container*. CTO.ai Blog | Cloud Native DevOps Workflows for Development Teams in Slack. https://cto.ai/blog/docker-image-vs-container-vs-dockerfile/

Blue, J. (2022, 30 maart). Creating a private container registry: repository and web service. *Medium*. https://medium.com/clarusway/creating-a-private-container-registry-repository-and-web-service-8c753b54f55c

*Use volumes*. (2020, 5 juni). Docker Documentation. https://docker-docs.uclv.cu/storage/volumes/