whitehorses

member of the human network

WHITEBOOK

Docker

November 2019

Auteur:
Maurik-Jan Veenman
INTEGRATIESPECIALIST

Inleiding

Op Oracle Open World 2017 bezocht ik een presentatie over Docker en Kubernetes. Ik zag nog niet wat je daar mee zou kunnen, maar er was duidelijk iets geweldigs aan de gang. In de jaren daarna kwamen deze termen toch regelmatig voorbij. Dat maakte mij nieuwsgierig genoeg om eens uit te zoeken wat het nu is, en wat ik eraan zou kunnen hebben. Hierbij mijn bevindingen.

Wat is Docker?

Docker biedt de mogelijkheid om op een Windows of Linux host computer een soort mini virtual machine (container) voor een dedicated oplossing te draaien. Je zou bijvoorbeeld Jenkins of een zelfgebouwde Microservice als Docker container kunnen draaien.

Bij een klassieke virtual machine oplossing wordt een Hypervisor als VMWare of OracleVM geïnstalleerd op het hostsysteem. Dit is een abstractie laag tussen de host en de Virtuele machine. Op zo'n Hypervisor kan dan een gast Operating System geïnstalleerd worden, waarin applicaties kunnen draaien, zo'n gast OS wordt dan een virtuele machine genoemd. Je hebt dan naast je gewenste applicatie wel als extra's ook een Hypervisor en het gast-OS nodig. Hierdoor wordt de grootte van een VM al snel enige GigaBytes , terwijl een Docker image gemeten wordt in MegaBytes.

Bij Docker wordt in plaats van een hypervisor gebruik gemaakt van twee basisfuncties van de Linux-kernel: Control groups (Cgroups) en kernel-namespaces. Cgroups beperken de toegang van processen op resources zoals het geheugen, CPU en I/O-poorten en voorkomen zo dat het proces zoveel resources gebruikt, dat andere lopende processen worden gehinderd. Namespaces zorgen ervoor dat een proces en de subprocessen daarvan worden beperkt tot een bepaald gedeelte van het gebruikte systeem.

Met bovenstaande technieken kunnen processen van Docker geïsoleerd worden. Deze techniek is tot versie 0.8.1 gebaseerd op Linux-containers (LXC). Sinds versie 0.9 hebben gebruikers toegang tot het zelfontwikkelde formaat 'Libcontainer'. Hierdoor is het mogelijk om Docker op verschillende platformen te gebruiken en één container op verschillende hostsystemen te laten draaien. Daardoor kunnen er ook Docker-versies voor Windows en MacOS worden aangeboden.

Waarom zou je Docker willen gebruiken?

Vanwege het feit dat je heel lichtgewicht diverse toepassingen kunt draaien op één host, en je zeker weet dat de container die je hebt gemaakt op je oplever systeem gaat werken, omdat alle benodigdheden ingebakken zitten.

Wat kunnen wij (Middleware specialisten) ermee?

Je kunt werkelijk alles in de container draaien, maar toepassingen die het voor ons interessant maken zijn microservices en CI/CD (Continuous Integration and Continuous Delivery) tools. Door de nieuwe Docker techniek kun je op één host al deze software, indien gewenst onder elk type OS, draaien en updaten.

Wat heb je nodig om met Docker aan de slag te gaan?

Installeer GitHub op je systeem. Importeer daarmee Docker en andere applicaties die je wilt gaan gebruiken, en je kunt met bovengenoemde statements aan de slag. Op de site van Docker staat een leuke tutorial om te leren werken met Docker (waar ook de installatie van GitHub en Docker wordt uitgelegd), en als je vervolgens googelt op termen als 'run python on docker windows' (of Linux) dan vind je allerlei voorbeelden om dat voor elkaar te krijgen.

Waaruit bestaat Docker?

Docker biedt de volgende vier belangrijkste tools voor haar functionaliteit:

- **Docker Engine;** die wordt gebruikt om Docker images van ofwel een Docker file ofwel een docker-compose.yml bestand te 'bouwen' en 'uit te voeren'. Als iemand een Docker commando uitvoert via de Docker Command Line Interface (CLI), dan praat hij tegen deze machine en vertelt hij hem wat hij moet doen.
- **Docker Compose;** deze tool gebruik je wanneer je een applicatie hebt die bestaat uit meerdere microservices, databases en andere subsysteempjes die met elkaar samenwerken. De docker-compose.yml stelt je in staat om deze onderdelen op een plek te configureren en ze allemaal in een keer te starten met één commando.
- **Docker Machine;** met deze tool kun je de Docker engine op virtual hosts installeren en de host besturen met docker-machine commando's. Omdat Docker images stabieler zijn geworden op de host platforms is de tool wat naar de achtergrond geschoven.
- **Docker Swarm;** wordt gebruikt om applicatieservices te deployen waar geen orchestratie voor nodig is, dus alleen een middel om in een keer vele applicaties te deployen en starten.

Images en Containers

De termen Docker image en Docker container worden nog wel eens door elkaar heen gebruikt. Het zijn echter twee verschillende dingen.

Docker **images** zijn executable packages waar alles in zit om een bepaalde applicatie uit te voeren: de code, libraries, omgevingsvariabelen en configuratiefiles.

Docker **containers** zijn runtime voorkomens van een Docker Image. Dus zo gauw een image uitgevoerd wordt en zich in het geheugen van de host computer bevindt, wordt het een container.

Hoe ziet het er nu uit: De Docker file

Welke commando's worden in Docker files gebruikt?

- `FROM` - dit initialiseert een nieuwe 'build stage' en zet de basis 'image' voor vervolginstructies. Een Docker file begint dus altijd met dit commando.
- `RUN` - voert commando's uit in een nieuwe laag in de image en 'commit' de resultaten. De veranderde image zal gebruikt worden door de volgende stap in de Docker file.
- `ENV` - zet de environment variable `<key>` naar de waarde `<value>`. Deze waarde blijft bestaan gedurende alle vervolgopdrachten in de 'build stage' en kan ook een andere waarde gegeven worden in vervolgstappen.
- `EXPOSE` - informeert Docker dat de container op een specifieke netwerkpoort gaat luisteren als hij eenmaal draait. Je kunt specificeren of de poort naar TCP of UDP luistert. Default is het TCP als niets wordt opgegeven. Hierdoor wordt de Docker container benaderbaar voor de host computer en de buitenwereld.
- `VOLUME` - creëert een mount point met de gespecificeerde naam en definieert het als een gemounte schijf van de host of eventuele andere containers.

Voorbeeld Dockerfiles:

Java Dockerfile (Linux)

```
# creëer een laag van de openjdk:8-jdk-alpine Docker image
FROM openjdk:8-jdk-alpine

# creëer de directory vanaf waar Tomcat zijn werkdirectories
maakt
VOLUME /tmp

# kopieer de JAR-file naar de container en hernoem hem naar
‹app.jar›
COPY build/libs /app

# voer de JAR uit op genoemd entrypoint
ENTRYPOINT [«java», «-Djava.security.egd=file:/dev/./
urandom», «-jar», «/app/java-example.jar»]
```

Python Dockerfile (Windows)

```
# creëert een laag van de python:alpine3.7 Docker image
FROM python:alpine3.7

# voegt files toe van de huidige directory van de Docker
client en stelt de # werkdirectory in
COPY . /app
WORKDIR /app

# creëert de applicatie met pip, gebruik makend van de
'requirements.txt'
# hierin staat alleen de tekst 'flask', omdat die applicatie
benodigd is
RUN pip install -r requirements.txt

# toon de applicatie aan de buitenwereld op poort 5000
EXPOSE 5000

# Met dit commando wordt python uitgevoerd in de container
CMD python ./index.py
```

Hoe run je de Docker files?

Met de volgende eenvoudige statements zet je de Docker images aan of uit:

`docker run my-image`
Het Docker run commando creëert een beschrijfbare containerlaag voor de gespecificeerde image, en start de image vervolgens.

`docker stop container-id`
Dit commando stopt de container met de naam ‘container-id’.

`docker ps`
Dit commando toont de containers die op dit moment draaien. Als je alle aanwezige containers wilt zien, gebruik dan de -a (of --all) toevoeging.

`docker rm`
Met dit commando verwijder je een of meerdere Docker containers.

Kubernetes

Kubernetes is platform dat in staat is om Docker containers te schalen en beheren over grote clusters, zo’n applicatie noemen we een orchestrator. Wanneer je een stel containers hebt draaien verdeeld over een aantal hosts, dan heb je zo’n orchestrator nodig om dingen in te stellen als: Waar gaat de volgende container starten? Hoe maak ik een container highly available? Hoe controleer je welke containers met andere containers kunnen communiceren? Kubernetes heeft daarnaast functies zoals *service naming and discovery, zodat je beschikbare services kunt vinden en hergebruiken, health monitoring, rolling updates (nieuwe versies implementeren terwijl de container gebruikt wordt) en horizontal autoscaling (horizontal scaling betekent het toevoegen van extra machines aan een cluster). Kubernetes kan overigens in een Docker container draaien. Over ‘eating your own dogfood’ gesproken!*

Concurrentie

Naast Docker zijn er nog een aantal applicaties die eraan gelieerd zijn. Ik bespreek even kort wat het inhoudt en wat je ermee kan:

- Joyent (triton) hebben een soort ‘elastic bare metal’ VM-omgeving, waarin uiteindelijk Docker wordt gedraaid om containers te runnen. Niet echt een nieuwe optie dus.
- Mesosphere is een omgeving om juist weer diverse Kubernetes omgevingen centraal te kunnen beheren, omdat dat bij grote aantallen zeer arbeidsintensief kan worden.
- Pivotal Software-VMware
  - Pivotal Container Service is een omgeving om Kubernetes in te draaien, ook weer Container management dus, net als Mesosphere.
  - Pivotal Build Service is een omgeving waarmee containers gemaakt kunnen worden à la Docker, bedoeld om te draaien binnen een Kubernetes omgeving.
- Rancher Labs, hiermee kan Kubernetes beheerd worden en containers gedraaid worden, ook weer op Enterprise niveau.
- Red Hat Openshift Red Hat heeft ervoor gekozen om een eigen applicatie te maken om containers te maken en te runnen, deze heet Openshift. Zij maken hierbij wel ook weer gebruik van Kubernetes.

Conclusie

Docker is een mooi middel om op een efficiënte wijze (eigen) software te draaien binnen een organisatie. Er is heel veel minder machinerie voor nodig dan bij VM's. Men kan snel mee veranderen bij veranderingen in de markt door andere software te gaan gebruiken zonder dat daar het machinepark erg voor aangepast hoeft te worden (ook al geldt dat, met wat meer moeite, ook voor VM's). Als ook Kubernetes gebruikt wordt is het onderhoudsvriendelijk doordat de software en het OS gepatcht kunnen worden terwijl de container draait.

Kortom, een bijzonder interessante optie voor organisaties die veel dedicated software gebruiken voor grote groepen gebruikers.

Bronnen en verder lezen

https://www.docker.com/
https://www.infoworld.com/article/3310941/why-you-should-use-docker-and-containers.html
https://hub.docker.com/search/?source=verified&type=image&page=3 (containers voor op Docker)