# Docker

## Optimaliseer het continuous delivery proces

Tegenwoordig is continuous delivery erg populair en applicaties worden automatisch uitgerold over verschillende omgevingen. Vaak blijft het daar echter bij en wordt de rest van de software (zoals de applicatieserver) handmatig of met een los proces uitgerold. Idealiter zouden applicaties inclusief alle benodigdheden uitgerold moeten worden. Dit is op te lossen door alle software op te nemen in Docker containers en die vervolgens met bijvoorbeeld Jenkins uit te rollen. Dit artikel geeft een introductie in Docker. Vervolgens worden de belangrijkste onderdelen behandeld met een aantal voorbeelden. Na het lezen van dit artikel kun je direct zelf aan de slag met Docker! Tevens wordt nog een vergelijking gemaakt tussen Docker en virtuele machines. Als laatste worden enkele toepassingen van Docker toegelicht.

**Johan Janssen** is Java ontwikkelaar bij Info Support

### Introductie

Docker kan gebruikt worden om applicaties te draaien binnen containers. Deze containers kunnen vervolgens op allerlei platformen uitgerold worden, zoals in figuur 1 te zien is. Docker containers maken gebruik van low level Linux kernel features, zoals cgroups. Containers kunnen het beste vergeleken worden met virtuele machines. Verderop in dit artikel is te zien dat containers in veel gevallen een beter alternatief zijn voor virtuele machines. Docker is een open source project, waarvan in maart 2013 de eerste release uitkwam. Al snel werd Docker erg populair en in juni 2014 kwam versie 1.0 uit. Deze versie is geschikt voor gebruik in productie-omgevingen. Docker Inc. - het bedrijf achter Docker - biedt ook betaalde ondersteuning aan. Op dit moment draait Docker alleen op een Linux kernel. Wel is het mogelijk om Docker in Windows of OS X te gebruiken door middel van Boot2docker of een virtuele machine. Microsoft heeft onlangs aangekondigd dat het Docker gaat ondersteunen in Windows Server.
Wat nog handig is om te weten, is het verschil tussen een image en een container in Docker-terminologie. Deze termen worden namelijk regelmatig verwisseld. Kort door de bocht is een container een draaiende instantie van een image. Het is mogelijk om meerdere containers te draaien op basis van hetzelfde image.

### Hello Docker

Het installeren van Docker is gemakkelijk en voor de meeste Linux-distributies is Docker standaard beschikbaar. In Ubuntu bijvoorbeeld is het commando

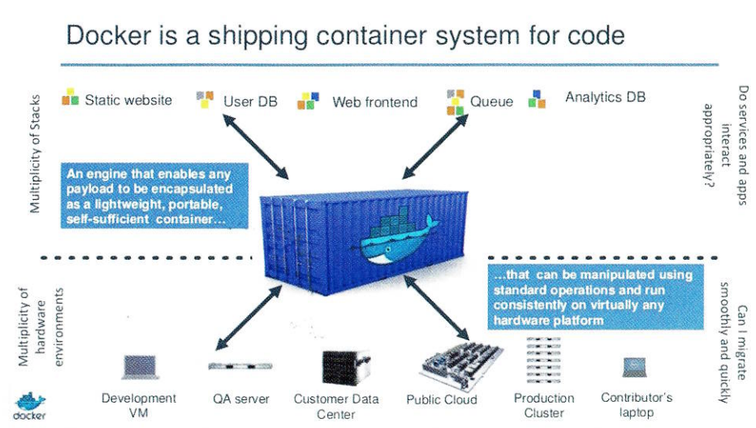
```
apt-get install docker.io
```

voldoende om Docker te installeren. Ubuntu heeft voor de naam 'docker.io' gekozen, omdat er al een ander pakket met de naam 'docker' bestaat. Voer vervolgens het volgende commando uit:

```
docker.io run -i -t ubuntu:saucy /bin/bash
```

Dit commando start de bash shell in een container, gebaseerd op Ubuntu Saucy. De opties '-i' en '-t' zorgen ervoor dat je een interactieve connectie en een terminal krijgt. Op deze manier is het mogelijk om de container te gebruiken en verder in te richten. Deze opzet is leuk om even snel te spelen met Docker. Voor serieuze toepassingen is het beter om gebruik te maken van Dockerfiles.

**DOCKER KAN HELPEN OM HET CONTINUOUS DELIVERY PROCES VERDER TE OPTIMALISEREN**




*Figuur 1 Software losgekoppeld van hardware*

### Dockerfiles

Hieronder is een gedeelte van een Dockerfile te zien. Met 'FROM' wordt aangegeven welk image als basis gebruikt wordt. Met 'RUN' kunnen standaard Linux-commando's uitgevoerd worden en met 'ENV' kunnen omgevingsvariabelen gezet worden. Met een beetje kennis van Linux en de voorbeeld Dockerfiles van internet is het eenvoudig om eigen containers te maken.

```
FROM ubuntu:saucy
...
RUN apt-get -y install oracle-java7-installer
ENV JAVA_HOME /usr/lib/jvm/java-7-oracle
```

Vervolgens kan het image gebouwd worden met:

```
docker.io build -t GeneralBase .
```

Dit commando downloadt het Ubuntu Saucy image, installeert Java en zet de omgevingsvariabele. Het uiteindelijke image krijgt de naam 'GeneralBase'. Het starten van een container op basis van dit image is niet nodig, aangezien we dit image als basis voor andere images gaan gebruiken.

### Containers stapelen

De functionaliteit uit het basis-image 'GeneralBase' kan gebruikt worden voor andere images. Bijvoorbeeld om images voor Jenkins, Nexus en SonarQube te maken, die allemaal Java nodig hebben. Als de Jenkins, Nexus en SonarQube containers op één machine draaien, dan is slechts één keer de schijfruimte nodig van 'GeneralBase'.
Hieronder staat een gedeelte van de Dockerfile voor SonarQube, die gebruik maakt van het basisimage 'GeneralBase'.

```
FROM GeneralBase
....
RUN unzip sonarqube-4.x.zip -d /opt
EXPOSE 9000
EXPOSE 9092
CMD ["/opt/sonarqube-4.x/bin/linux-x86-64/sonar.sh", "console", "/bin/bash"]
```

Nieuw in deze Dockerfile is 'EXPOSE', waarmee een poort vanuit de container opengezet kan worden. Met 'CMD' wordt een commando uitgevoerd. In dit geval wordt SonarQube gestart. Met

```
docker.io build -t SonarQube
```

wordt het image, genaamd SonarQube, gebouwd op basis van bovenstaande Dockerfile. Door middel van het commando:

```
docker.io run -p 9000:9000 -p 9092:9092 -d SonarQube
```

wordt de container gestart. Met '-p' kan een portmapping opgegeven worden in het formaat 'x:y'. De 'x' staat voor de poort, die de container krijgt binnen het host OS. De 'y' is de poort binnen de container, die met 'EXPOSE' in de Dockerfile werd opengezet. Deze mapping wordt tijdens het starten van de container gedaan. Hierdoor is het mogelijk om een Dockerfile te maken voor bijvoorbeeld Tomcat, die naar poort 8080 luistert. Vervolgens kunnen meerdere containers op basis van dezelfde Dockerfile gemaakt worden. Bij het starten van de Tomcat containers kunnen verschillende poortnummers (8081, 8082...) opgegeven worden voor het host OS.

## Data volumes

Docker wordt gebruikt om functionaliteit te isoleren. Het is de bedoeling dat iedere applicatie in een losse container draait. Hierdoor zijn applicaties makkelijker te upgraden en zitten ze elkaar niet in de weg. Naast het scheiden van functionaliteit zou ook de data van de applicatie gescheiden moeten worden. Dit is mogelijk door de data en de applicatie in losse containers te draaien. Alle data die opgeslagen en bewaard moet worden, komt in een losse datacontainer. Het is voldoende om deze datavolumes te backuppen. De andere containers kunnen zo weer opnieuw opgestart worden. Het backuppen dient wel ingericht te worden, want Docker doet dit niet automatisch. Hieronder worden de stappen beschreven om een datavolume voor Jenkins op te zetten. Het voorbeeld gaat er vanuit dat zowel de Jenkins container als het datavolume (JenkinsData-Container) op dezelfde machine staan. In de laatste stap worden de applicatiecontainer en het datavolume aan elkaar gekoppeld.

- Specificeer de JENKINS_HOME locatie in de Dockerfile

```
ENV JENKINS_HOME /var/JenkinsData
```

- Creëer een datavolume met daarin de /var/JenkinsData folder

```
docker.io run -v /var/JenkinsData --name
JenkinsDataContainer ubuntu:saucy true
```

- Bouw het Jenkins image op basis van de Dockerfile

```
sudo docker.io build -t Jenkins .
```

- Geef het datavolume mee tijdens het starten van de Jenkins container

```
docker.io run -p 8080:8080 --volumes-
from JenkinsDataContainer -d Jenkins
```

## Registry

Om images op te slaan en te verspreiden, is het aan te raden om van een registry gebruik te maken. Hierdoor kan dezelfde image op bijvoorbeeld een laptop en een server gestart worden. Ook is het mogelijk om bij problemen de hele omgeving opnieuw uit te rollen door de images uit de registry te halen. Een Docker-registry lijkt een beetje op een Git-repository. Er is een publieke instantie beschikbaar, maar het is ook mogelijk om een eigen registry te maken. Dat kan eenvoudig door een nieuwe container te starten, die gebaseerd is op het registry image met:

```
docker.io run -p 5000:5000 registry
```

In **Figuur 2** is te zien hoe Dockerfiles, images, containers en de registry samenhangen. Van een Dockerfile kan een image gebouwd worden. Op basis van een image kan een container gestart worden. De container kan vervolgens weer als image opgeslagen worden. Images worden met push naar de Docker-registry gestuurd. Vanaf een andere machine kan vervolgens het image binnengehaald worden, door middel van de commando's search en pull. Vervolgens kan met run de container gestart worden, die gebaseerd is op het image.

## Opslaan van (gewijzigde) images

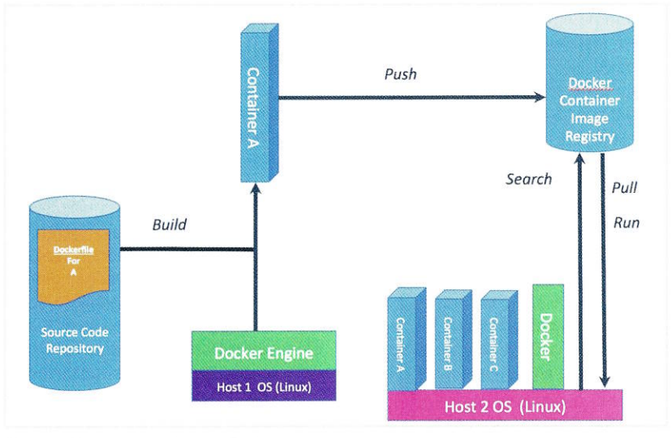
Het opslaan van een (gewijzigd) image in de registry werkt ook ongeveer hetzelfde als met Git. Voor onderstaand voorbeeld worden twee machines gebruikt. Beide machines bevatten 'version-0.1' van een image. Op machine één wordt 'version-0.2' van de image gemaakt op basis van 'version-0.1' en deze wordt via de registry binnengehaald op machine twee. De volgende stappen worden op machine één uitgevoerd:

- Sla de container op

```
docker.io commit [containerid] [registryserver]:5000/ test-version-0.2
```

- Stuur het image naar de registry

```
docker.io push [registryserver]:5000/ test-version-0.2
```




*Figuur 2 Algemene werking van Docker*

**DOORDAT DOCKER WEINIG OVERHEAD HEEFT, IS HET PRIMA GESCHIKT VOOR MINDER KRACHTIGE MACHINES**

De volgende stappen worden op machine twee uitgevoerd om 'version-0.2' binnen te halen:

- Haal het image op uit de registry

```
docker pull [registryserver]:5000/
test-version-0.2
```

- Toon de beschikbare images

```
docker images -tree
```

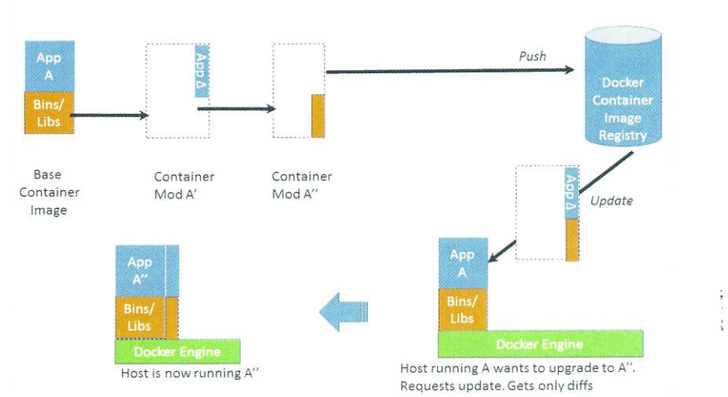
- Het resultaat is als volgt

```
└─153b 194.2 MB test-version-0.1:latest
  └─ff7e 194.2 MB test-version-0.2:latest
```

- Start de container

```
docker.io run -i -t ff7e /bin/bash
```

Beide images staan nu op beide machines. Ook hebben de images dezelfde ID (153b, ff7e) op beide machines. Tevens is te zien dat de 'version-0.2' image op de 'version-0.1' image gebaseerd is en dat de schijfruimte minder dan een MB is. Dat is mogelijk, doordat alleen de diffs opgeslagen en gedownload worden. In **figuur 3** is nogmaals te zien hoe container- en image-updates werken. Behalve de applicatiewijzigingen worden ook de overige containerwijzigingen (bijvoorbeeld Linux security updates) als diffs opgeslagen in de registry.




*Figuur 3 Wijzigen van een Docker container*

## Virtuele machines

Docker kan als alternatief voor virtuele machines gebruikt worden of binnen een virtuele machine. Het voordeel van Docker is dat rechtstreeks gebruik gemaakt wordt van de host OS. Hierdoor is de overhead van Docker minimaal en de performance bijna gelijk aan het draaien op fysieke hardware. Zelf hebben we Docker gebruikt om een buildstraat en OTAP-straat te maken met losse containers voor Jenkins, Nexus, GitBlit, SonarQube en vier Tomcat applicatieservers. De containers bevatten ook de basis Ubuntu-image en Java, maar misten een aantal Jenkins plugins die we later hebben toegevoegd. In totaal was hiervoor ongeveer 900 MB schijfruimte voor nodig. Het downloaden (via een 30Mbit-lijn) en het starten van de containers duurde minder dan 4:15 op een laptop. Als een container al gebouwd is, dan wordt die binnen een aantal milliseconden gestart. Probeer dat maar eens met virtuele machines!

## Toepassingen

Docker kan gebruikt worden op praktisch alle apparaten met een Linux kernel. Doordat Docker weinig overhead heeft, is het ook prima geschikt voor minder krachtige machines. Zelf heb ik Docker (inclusief Tomcat en een applicatie) draaien op een Raspberry Pi. Binnen ons bedrijf draaien we een applicatie die gebruik maakt van Docker in de cloud. Voor het deployen van die applicatie is een continuous delivery pipeline opgezet met Docker containers. Deze is gebaseerd op Jenkins, Nexus, GitBlit, SonarQube en de Build Pipeline Plugin en Publish Over SSH Plugin voor Jenkins. Dezelfde containers die in de cloud draaien, kunnen ook op andere plekken gestart worden, zoals op een laptop. Let er wel op dat containers niet uitwisselbaar zijn tussen verschillende hardware-architecturen. Een container die op de Raspberry Pi met ARM-architectuur draait, zal niet op je laptop draaien, tenzij die ook een ARM-processor heeft.

## Conclusie

Docker kan helpen om het continuous delivery proces verder te optimaliseren. Het is erg eenvoudig om met Docker te starten en het werkt erg prettig. Het is niet voor niets dat Docker op dit moment één van de populairste projecten is. Meer informatie, inclusief de complete Dockerfiles en meer uitleg over de toepassingen, is te vinden op **http://blogs.infosupport.com/author/johanj/**. ■