

docker

Onderzoek Docker

Auteurs: Malcolm Kindermans & Maurits van Mastrigt

Datum: 7 januari 2015

INLEIDING

Aanleiding

De opkomst van Docker is voor veel systeemadministratoren haast niet te missen. Na de eerste demonstratie van Solomon Hykes¹ in maart 2013 ontploft er een bom in de wereld van virtualisatietechnieken. Veel bedrijven zien de potentie van de technologie en gaan er dan ook direct mee aan de slag.

Peperzaken heeft onlangs een artikel² geschreven over hun keuze om over te stappen naar Docker. In dit artikel wordt uitgelegd hoe Peperzaken een alternatief zocht om van hun problemen met het uitrollen van software af te komen en om de ontwikkelomgevingen en -workflow consistent te krijgen.

De snelle groei van Docker laat zien dat het een probleem oplost die veel ontwikkelaars hebben. De vraag is alleen: welk probleem is dit precies? En waarom zijn er zoveel (grote) partijen geïnteresseerd in deze technologie? Deze vragen zijn de aanleiding voor dit onderzoek.

Context

Dit onderzoek is gedaan in de context van het ontwikkelen van onderzoeksvaardigheden vanuit school, in het kader van het thema Software Engineering. Van een HBO student wordt verwacht dat deze zelfstandig onderzoek kan verrichten en daarmee zijn beroepsproducten kan onderbouwen. In tegenstelling tot de universiteit, behoort bij het HBO het doen van onderzoek op zich niet tot het curriculum. Op het HBO heeft het onderzoekend vermogen van de student een ondersteunende rol bij het onderbouwen van (goede) beroepsproducten.

Probleem

Bij veel projecten ontstaat hetzelfde probleem: bij een deploy van de lokale ontwikkelomgeving naar de testomgeving werkt het "ineens" niet meer. Een reactie die dan vaak gehoord wordt, is "Ik snap het niet, bij mij werkt het gewoon." Dat komt doordat de omgeving die lokaal wordt gebruikt, vaak anders is dan de omgeving op de server. Dat kan verschillende oorzaken hebben. Bijvoorbeeld: het verschil in besturingssysteem tussen developers kan het noodzakelijk maken dat er verschillende libraries gebruikt worden. Dit verschil in libraries kan voor "onverklaarbare" problemen zorgen.

Daarnaast kan het schakelen tussen verschillende projecten ook een problemen veroorzaken. Het ene project draait bijvoorbeeld op een server met Ubuntu 14.04, PHP 5.3 en MySQL 3.23, terwijl het andere project draait op een server met CentOS 6, PHP 5.5 en MySQL 5.5.36. Wanneer de lokale ontwikkelomgeving dan is ingericht op het project dat PHP 5.5 gebruikt, kunnen er problemen ontstaan bij het deployen naar de productieomgeving. Ditzelfde probleem kan zich ook voordoen tussen verschillende developers.

Docker lost dat probleem op. Docker maakt het heel eenvoudig om een omgeving op te zetten. Het maakt daarbij niet uit of het om een ontwikkelomgeving of een productieomgeving gaat. Docker biedt zelfs de mogelijkheid om op de pc van één developer, meerdere omgevingen te kunnen laten draaien. Wanneer op de server een update wordt gedaan van PHP 5.3 naar PHP 5.4, kan lokaal door iedere developer Docker worden geüpdatet, zodat de omgevingen weer identiek zijn.

Waarom is Docker relevant?

Het inzetten van deze nieuwe technologie heeft veel impact in de wereld van virtualisering. Zo zijn er zowel grote - Google, Microsoft, Amazon - als kleine bedrijven - Peperzaken, Innovatio - geïnteresseerd in de technologie.

Docker tackelt een specifiek probleem. Voordat Docker er was hebben veel bedrijven hun eigen manier bedacht om dit probleem op te lossen, maar Docker plaatst elk stukje van de applicatie in een aparte "container" (denk hierbij aan de universele scheepscontainers). Hierdoor kan een container worden verplaatst naar bijvoorbeeld een andere server, zonder dat er ook maar iets aan de applicatie hoeft te veranderen. De technologie van Docker is slim genoeg om zelf uit te vogelen waar de "container" op moet draaien (bijvoorbeeld wat voor soort cloud server het is) en past de container - de schil - aan naar de onderliggende hardware zonder dat het stukje van de applicatie in de container dit merkt. Hierdoor zal de applicatie blijven werken, ongeacht de onderliggende server hardware.⁵

De oplossing die Docker biedt is relevant voor bedrijven zoals Google, Ebay etc., omdat cloud computing nu nog efficiënter kan worden ingezet. De containers van Docker kunnen eenvoudig worden opgezet en draaien volledig zelfstandig. Hierdoor is het schalen van applicaties eenvoudiger en kan er beter worden opgegaan met bijvoorbeeld uitval van servers - er worden simpelweg nieuwe containers gestart op een andere server.⁶

Omdat Docker pas sinds juli 2014 officieel gereleased is, is het opmerkelijk dat er al zoveel bedrijven zijn die de software adopteren in hun ontwikkelproces.⁷

Stakeholders

Een groot deel van de bedrijven gebruikt inmiddels virtualisering. Ook bedrijven die de hosting van hun server uitbesteden, zullen hun software zeer waarschijnlijk in een Virtuele Machine (VM) hebben draaien. Dit onderzoek is bedoeld voor bedrijven die virtualiseringstechnieken gebruiken en een globaal overzicht willen hebben van Docker.

Tevens is dit onderzoek bedoeld voor ontwikkelaars, om de stap naar het gebruik van Docker kleiner te maken. Deze stap wordt verkleind door beter inzichtelijk te maken wat Docker is, waarvoor het wordt gebruikt, wie het gebruiken en wat het voor een bedrijf kan betekenen.

De opzet van het onderzoek

Het onderzoek is gedaan in de vorm van een literatuurstudie in combinatie met een kwalitatief onderzoek. Zo wordt er in dit document veel materiaal samengevat om de lezer een goed overzicht te geven van de verschillende aspecten van Docker. Tevens is er gekeken naar de manier waarop Docker wordt ingezet. Dit wordt gedaan door middel van online materiaal en in de vorm van een demonstratie opstelling (testomgeving). De opstelling is bedoeld om een beter gevoel te krijgen over hoe de software functioneert, zodat hierover een mening kan worden gevormd. De resultaten hiervan zullen vervolgens worden gepresenteerd aan de lezers.

Dit document behandelt de volgende aspecten:

- Wat is Docker?
- Waarom Docker?
- Architectuur van Docker.
- Waar past Docker in het ontwikkelproces?
- Wat is de maturity van Docker en de community er omheen?
- Wat is de leercurve van Docker?
- Voor wat voor soort bedrijven is Docker relevant?
- Welke bedrijven gebruiken Docker?

De tijdsperiode waarin dit onderzoek is verricht is van 15 december 2014 tot en met 6 januari 2015. De onderzoekers zijn Malcolm Kindermans en Maurits van Mastrigt, beide vierdejaars Informatica studenten aan de Hanzehogeschool in Groningen.

Inleiding	2
Aanleiding	2
Context	2
Probleem	2
Waarom is Docker relevant?	3
Stakeholders	3
De opzet van het onderzoek	4
Doelstelling	7
Literatuuronderzoek	8
Wat is Docker?	8
Hoe werkt Docker?	8
Waarom Docker?	10
Voordelen	10
Nadelen	10
Alternatieven	11
Architectuur van Docker	12
Waar past Docker in het ontwikkelproces?	14
Alternatieven	15
Gedistribueerde componenten	15
Wat is de maturity van Docker en de community er omheen?	16
Conclusie	17
Wat is de leercurve van Docker?	18
Voor wat voor soort bedrijven is Docker relevant?	20
Welke bedrijven gebruiken Docker?	22
Spotify	22
Ebay	22
Gilt	22
Yelp	23

Bleacher Report	23
Baidu	23
Conclusie	24
Testconfiguratie	25
Installatie	25
Opzet testomgeving	27
Resultaten	29
Stappen	29
Overig	31
Bevindingen	34
Opzetten van de container	34
IP-adres van Docker	34
NGinX en HHVM	34
Koppelen van componenten	35
Grote MySQL container	35
Conclusie	36
Discussie	37
Bronnen	38

DOELSTELLING

Ondanks de toenemende populariteit van Docker is het voor bedrijven van belang om wel afgewogen keuzes te maken in het kiezen van bepaalde technieken. Het inzetten van Docker brengt bepaalde complicaties met zich mee, waaronder het omzetten van de infrastructuur naar losgekoppelde softwarecomponenten. Tevens moet er rekening worden gehouden met het migratietraject, de kennisinvestering, de community (voor hulp bij het oplossen van problemen) en de veiligheid van het systeem. Ook kan informatie over het gebruik van Docker bij andere bedrijven meer inzichten bieden in bijvoorbeeld de voor- en nadelen van een Docker infrastructuur.

Dit onderzoek geeft inzage in de verschillende aspecten waar rekening mee gehouden dient te worden bij het maken van de keuze voor Docker. Deze literatuurstudie tracht antwoord te geven op de vraag **'Wanneer biedt Docker toegevoegde waarde?'**

LITERATUURONDERZOEK

Wat is Docker?

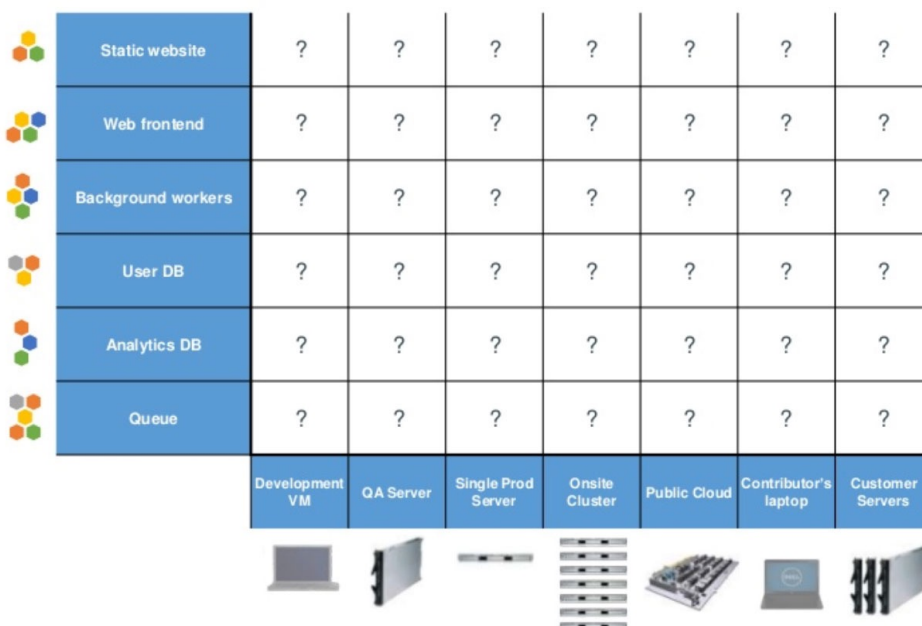
Docker is een open platform waarmee gedistribueerde applicaties kunnen worden gebouwd, getransporteerd en gerunt. Eén van de kenmerken van Docker is dat applicaties snel in elkaar kunnen worden gezet. Door gebruik van Docker kan de IT-afdeling binnen een bedrijf gemakkelijk overal dezelfde omgeving en apps draaien.

Docker bestaat uit twee 'onderdelen', namelijk de Docker Engine en de Docker Hub. De Docker Engine is een lichtgewicht packaging tool die makkelijk kan worden verplaatst. De Docker Hub is een cloud service die gebruikt wordt voor het delen van applicaties en automatiseren van workflows.







HOE WERKT DOCKER?

Er wordt steeds meer verschillende software gebruikt, met verschillen versies en op verschillende hardware. Dit heeft geresulteerd in wat de founder van Docker (Solomon Hykes) de 'Matrix of Hell' noemt:

The Matrix of Hell



The diagram illustrates 'The Matrix of Hell', a metaphor for the complexity of running different applications across various hardware environments. It consists of a grid where rows represent application types and columns represent hardware environments. Each cell contains a question mark, indicating that any application can run on any hardware, leading to a complex matrix of possibilities.

	Static website	?	?	?	?	?	?	?
	Web frontend	?	?	?	?	?	?	?
	Background workers	?	?	?	?	?	?	?
	User DB	?	?	?	?	?	?	?
	Analytics DB	?	?	?	?	?	?	?
	Queue	?	?	?	?	?	?	?
		Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers

Below the hardware environment labels, there are icons representing each environment: a laptop for Development VM, a server rack for QA Server, a single server for Single Prod Server, a cluster of servers for Onsite Cluster, a server rack for Public Cloud, a laptop for Contributor's laptop, and a server rack for Customer Servers.



Deze complexiteit kan worden vergeleken met het transporteren van goederen. Er is een grote verscheidenheid aan afmetingen van producten die moeten worden vervoert. Hierbij kan worden gedacht aan bijvoorbeeld olievaten, kratten en auto's. Deze drie producten hebben allerlei verschillende afmetingen en bij het vervoeren dient rekening te worden gehouden met verschillende aspecten, zoals breekbaarheid en ontvlambaarheid. Deze producten kunnen vervoerd via verschillende transportmiddelen, zoals per schip, vliegtuig of vrachtwagen. In de praktijk kan dit dus voor veel compatibiliteitsproblemen zorgen.

De oplossing die hiervoor bedacht is, is een standaard over de afmetingen van het product dat getransporteerd wordt: de container. Wanneer de verschillende software/libraries worden vergeleken met de producten die moeten worden getransporteerd en de verschillende hardware componenten worden vergeleken met de transportmiddelen, zien we veel overeenkomsten. Dit is waar Docker voor kan worden gebruikt. De verschillende software kunnen worden gedraaid in zogenaamde containers. Docker biedt vervolgens voor alle soorten hardware een interface om deze container te kunnen laten draaien. Hiermee is een standaard ontwikkeld die het mogelijk maakt om op iedere hardware dezelfde software te kunnen draaien zonder compatibiliteitsproblemen.

Dit kan eenvoudig worden bereikt door een *Dockerfile* te maken met daarin alle acties die moeten worden uitgevoerd, om zo een 'image' te maken. Een image is een read-only template, waarop bijvoorbeeld het besturingssysteem en de web applicaties staan geïnstalleerd. Met deze image kan vervolgens een container worden gemaakt. In het hoofdstuk over de *“Wat is de leercurve van Docker?”* wordt hier ook verder over gesproken.

Een Docker container kan vergeleken worden met een directory die alles bevat wat nodig is om een applicatie te kunnen draaien. Docker biedt de mogelijkheid om containers met elkaar te linken. Hierdoor ontstaat de mogelijkheid om bijvoorbeeld de database in een aparte container te laten draaien. Dit maakt de afhankelijkheden tussen de verschillende container minder complex. Hoe groter en dynamischer het systeem, hoe meer tijds winst dit oplevert.

Een image bestaat uit een aantal verschillende layers. Door gebruik te maken van het union file system¹³, worden deze verschillende layers gecombineerd tot één image. Hierin zit ook een van de redenen opgesloten dat Docker zo licht is. Wanneer iets aangepast wordt in één van de layers, hoeft de image niet in zijn geheel te worden verwijderd om vervolgens volledig te kunnen worden herbouwd. Enkel de nieuwe (of aangepaste) layer hoeft te worden toegevoegd (of aangepast).

Een image wordt opgebouwd door het uitvoeren van een aantal instructies. Elk van deze instructies bouwt een nieuwe layer. Deze instructies worden opgeslagen in de eerder genoemde *Dockerfile*.

Waarom Docker?

Deze sectie beschrijft de voor- en nadelen omtrent Docker. Ook wordt er gekeken naar mogelijke alternatieven.

VOORDELEN

Docker biedt een aantal duidelijke voordelen, ten opzichte van het traditionele model. Zo zijn Docker containers licht om op te zetten ('lightweight') en is de software binnen de container volledig afgesloten ('encapsulated'), tenzij anders aangegeven. Het grote voordeel van Docker is dat de containers met eenvoud op andere servers kunnen worden ingezet ('shippable'). Dit is mogelijk doordat elke container vanaf een aangegeven 'base image' draait, zoals Ubuntu 14.04 of een andere linux distributie.

De duidelijke isolatie van software binnen de containers leidt tot andere voordelen:

1. Potentie voor een hogere uptime:
 - Meer inzicht in falende componenten.
 - Falende onderdelen kunnen eenvoudig worden vervangen met een andere instantie van de container.
2. Eenvoudiger beheer van de individuele componenten:
 - Meer inzicht in prestaties, wat leidt tot betere support.
 - Eenvoudig upgraden van containers.
3. Decompositie van de applicatie zorgt voor meer eenvoud in het redeneren over de dienst.
4. De componenten communiceren enkel via opengestelde kanalen, waarmee de veiligheid wordt vergroot.

Verder draait de onderliggende techniek ('LXC')¹⁷ op een laag niveau, waardoor overhead laag is en Docker snel blijft. Ook biedt Docker de mogelijkheid om images te delen via de Docker Hub. Waardoor de images naderhand in zijn geheel kunnen worden binnengehaald, wat het (relatief lange) proces van het opzetten elimineert. Met deze images kunnen vervolgens containers worden gestart.

NADELEN

Het opdelen van de applicatie in verschillende componenten heeft veel voordelen, echter is deze ontwerpfilosofie wel verplicht. Indien de applicatie architectuur bestaat uit veel nauw samenwerkende componenten, wordt het opzetten en beheren van deze componenten erg complex en is Docker niet altijd de juiste oplossing. Zo heeft het weinig nut om de gehele applicatie in een enkele container te draaien, terwijl de software ook op de daadwerkelijke server gedraaid kan worden - ervan uitgaande dat de server enkel voor de applicatie wordt gebruikt. De isolatie kan soms meer tijd en moeite kosten dan dat het uiteindelijk oplevert.

Tevens maakt Docker het opzetten van de applicatie een stuk complexer, omdat er van te voren moet worden nagedacht over de infrastructuur en scheiding en koppeling van de componenten.

Ook draait Docker, vanwege de onderliggende techniek, alleen op Linux. Gelukkig worden veel applicaties op de dag van vandaag al op Linux servers gedraaid. Gebruik van Docker op Windows of Mac OS X is mogelijk met behulp van 'Boot2docker'¹⁸. Hiermee kan Docker schijnbaar direct worden aangesproken via de command line, echter worden de commando's doorgestuurd naar een virtuele machine waar Docker op draait.

Tot slot zijn er, op het moment van schrijven, een aantal problemen met de veiligheid van containers. Zo kan een container met administrator rechten de bovenliggende laag overnemen¹⁹. Dit kan de server zelf zijn, maar ook een andere Docker container (een container draait Linux, waarop Docker geïnstalleerd kan worden). Dit is een serieus beveiligingsprobleem, tenzij de container niet de juiste rechten heeft of er een container laag omheen wordt gezet (die op zijn beurt geen administrator rechten heeft) indien de administrator rechten zijn vereist.

Bij het binnenhalen van images wordt de authenticiteit van de image incorrect geverifieerd²⁰. Hierdoor kan een aanvaller misbruik maken van mogelijke fouten in de systeemtools die Docker gebruikt.

ALTERNATIEVEN

Naast Docker, en de handmatige ('traditionele') manier voor het opzetten van een server, zijn er genoeg alternatieven. In het kort, een aantal noemenswaardige alternatieven voor Docker:

1. **VMWare**²¹ en **VirtualBox**²²: Hosting binnen een virtuele machine, wat ongeveer gelijk staat aan de volledige applicatie binnen een enkele container.
2. **Vagrant**²³: Een hulpmiddel voor het automatisch opzetten van virtuele machines.
3. **Puppet**²⁴, **Chef**²⁵ en **Ansible**²⁶: Zogeheten 'provisioning tools', voor het configureren van een server (gebruikers, rechten, software, etc.) middels configuratiebestanden.
4. **Flockport**²⁷, **Rocket**²⁸ en **LXD**²⁹: Net als Docker: containers met behulp van LXC. Elk met een eigen ontwerpfilosofie:
 - Flockport: Gebruikersgemak en veel officiële containers, waardoor software eenvoudig is op te zetten.
 - Rocket: Veiligheid. Docker draait als één proces, waardoor deze niet met beperkte rechten gedraaid kan worden.
 - LXD: Een verbeterde gebruikersinterface voor LXC (tevens van de makers van LXC).

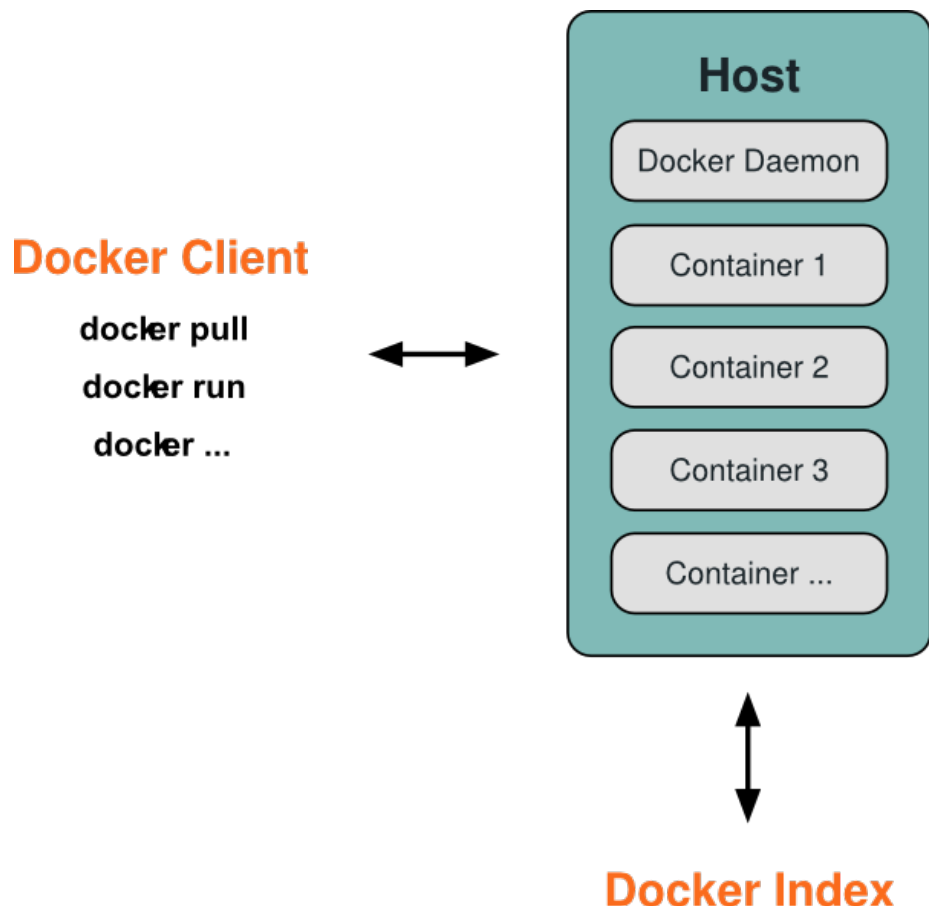
Architectuur van Docker

Voor het gebruik zijn de volgende dingen nodig:

- een computer of server met een Linux besturingssysteem (de 'host');
- de Docker software (waarin LXC is inbegrepen);
- optioneel: een internet verbinding voor het gebruik van Docker Hub.

Het concept van Docker is vrij eenvoudig. Er draait een applicatie op de achtergrond die de images en containers beheert. Deze applicatie wordt aangestuurd met behulp van een command line tool (zie ook het hoofdstuk "*Testconfiguratie*"). Tot slot is er de Docker Index, dit is de lijst met images die wordt aangeboden door de Docker Hub. De command line tool helpt ook bij het binnenhalen van images van de Docker Hub en het opslaan van eigen gemaakte images.

Het volgende figuur laat zien hoe Docker is opgebouwd:



De zogeheten 'Docker Daemon' is de centrale applicatie die in de achtergrond draait. Deze 'client-server' architectuur biedt een eindpunt alle zaken omtrent Docker worden aangestuurd. Dit stelt tools als *Boot2docker* in staat om van buitenaf te communiceren en Docker aan te sturen.

De command line tool kan op zijn beurt communiceren met dit eindpunt, om zo containers op te zetten (aan de hand van de images), te starten, te stoppen, te inspecteren etc. Verder kan deze tool images ophalen via de Docker Index. Zoals in de afbeelding te zien, gaat dit via de Docker Deamon. Tevens kunnen images worden opgeslagen in de Docker Hub.

Het publiceren van deze image betekend ook dat een ieder ander het kan hergebruiken. Het is ook mogelijk een eigen (privé) Docker register op te zetten, wat handig is bij het werken met gevoelige informatie en waarmee de huidige beveiligingsproblemen worden geëlimineerd.

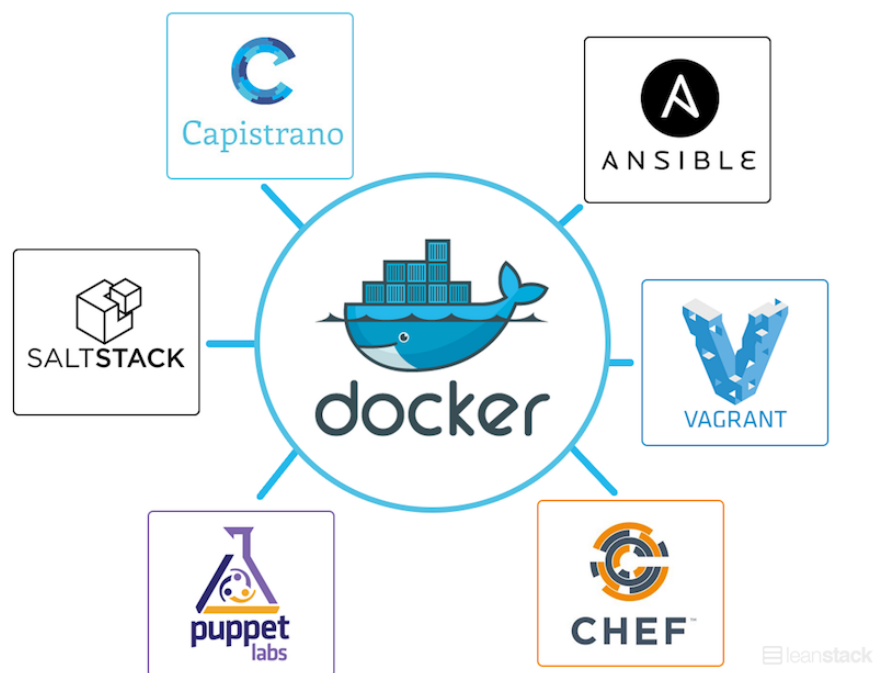
Waar past Docker in het ontwikkelproces?

Docker is een hulpmiddel voor systeembeheerders (ookwel 'devops' genoemd, beheerders van 'development' en 'operations') om de softwarecomponenten (delen van de infrastructuur) van een applicatie op een gestructureerde manier op te zetten en te beheren.

Het werk van een systeembeheerder is niet altijd makkelijk. Zo ontstaan er eenvoudig verschillen in de ontwikkelomgevingen van de ontwikkelaars - alleen al een verschillend besturingssysteem zorgt veelal voor problemen. Ook schaal een applicatie niet altijd goed, omdat de infrastructuur handmatig is opgezet, waarbij de samenhang als het ware 'opnieuw is uitgevonden'. Denk hierbij aan bijvoorbeeld het handmatig uitvoeren van de installatiestappen voor de software onderdelen.

Docker is een stap in de goede richting voor de volgende problemen of moeilijkheden:

- **Verschillen in ontwikkelomgeving:** Alle containers gebruiken dezelfde 'base image' en installatiestappen, hierdoor zullen deze - uitgezonderd van de gekoppelde data - niet verschillen.
- **Verschillen in aanpak:** Docker biedt een gestandaardiseerde en herhaalbare aanpak. Hierdoor kunnen images van andere worden hergebruikt, wat veel tijd scheelt. Doordat het wiel niet telkens opnieuw wordt uitgevonden, kan er ook meer tijd worden besteed aan het oplossen van beveiligingsproblemen en dergelijke.
- **Uitrollen van applicaties:** Het starten en stoppen van containers kan worden geautomatiseerd. Hierdoor hoeven er niet handmatig servers worden herstart en wordt er inzicht geboden in uitgeschakelde of falende componenten van de applicatie. Wat het mede mogelijk maakt om bij het falen van een component, automatisch een container te herstarten - eventueel op een backup server.
- **Het schalen van de applicatie:** Docker kan worden ingezet als een dienst, waarbij er met een druk op de knop meer containers worden gestart of gestopt. Zoals bijvoorbeeld de Kubernetes³⁷ dienst van Google.



ALTERNATIEVEN

Uiteraard is Docker niet de eerste die dit probleem probeert te tackelen. Zo zijn er veel andere softwarepakketten die de problemen elk op hun eigen manier aanpakken. Een veel gebruikte aanpak is het geautomatiseerd configureren van de volledige server. Bijvoorbeeld door gebruikers en services te persisteren (zorgen dat ze respectievelijk aanwezig zijn/goed geconfigureerd zijn of draaien). Chef is hier een voorbeeld van.

Gelukkig sluit Docker weinig van deze configuratiemiddelen uit. Docker zal eerder eenvoudiger te gebruiken zijn in combinatie met een zogeheten 'provision tool' (bijvoorbeeld Chef, Puppet of Ansible). In de onderstaande afbeelding is te zien welke tools Docker, op het moment van schrijven, al ondersteunen.

In plaats van dat Docker een 'one size fits all' oplossing aanbiedt, stelt het de gebruiker enkel in staat om software in een geïsoleerde omgeving te draaien. Hierdoor kan de stabiliteit van het opzetten of uitrollen van een applicatie juist worden vergroot. Zo kunnen 'provision tools' het bouwen van images (met behulp van de '*Dockerfile*' bestanden) en het (her)starten en stoppen van containers verzorgen.

Ook zijn er inmiddels hulpmiddelen beschikbaar om Docker in te zetten bij een infrastructuur met gedistribueerde componenten.

GEDISTRIBUEERDE COMPONENTEN

Omdat Docker geen bewustzijn heeft van andere servers en enkel de containers op de eigen server beheert, is er uiteraard behoefte aan een middel die de distributie van containers verzorgt. Denk hierbij aan het verdelen van de serverlast over meerdere servers. Bij bijvoorbeeld duizenden page requests naar een webserver is het wenselijk dat de werklast wordt verdeeld, om ervoor te zorgen dat de server niet overbelast raakt.

Ondanks de recente lancering zijn er al meerdere partijen die dit probleem oplossen:

1. Weave³⁸
2. Kubernetes³⁷
3. Core OS/Flannel³⁹
4. Pipework⁴⁰
5. SocketPlane⁴¹
6. Mezos⁴²

Deze softwarepakketten hebben ieder hun verschillen in aanpak. De keuze hangt hierin af van de eisen die aan de infrastructuur zijn gesteld.

Wat is de maturity van Docker en de community er omheen?

Docker is in maart 2013 gelanceerd en dus nog vrij jong. Daaruit zou de conclusie getrokken kunnen worden dat Docker nog veel kinderziektes heeft en nog niet klaar is voor de 'grote' markt. Toch vertellen de cijfers in de praktijk ons iets anders. Dit onderdeel zal wat dieper ingaan op de maturity (volwassenheid, mate van gevorderdheid van ontwikkeling) van Docker en de community er omheen.

Zoals in het hoofdstuk “*Welke bedrijven gebruiken Docker?*” blijkt, zijn er veel toonaangevende bedrijven die gebruik maken van Docker. Daarin is te zien dat partijen als Yelp, Spotify en Ebay gebruik maken van Docker. Dergelijke bedrijven maken geen gebruik van een platform dat nog niet ver genoeg ontwikkeld is om gebruikt te kunnen worden. Hieruit kan dus geconcludeerd worden dat Docker die mate van maturity heeft bereikt dat het gebruikt kan worden door grote bedrijven. Uit de statistieken van Alexa blijkt ook, dat zo'n 60 á 70% van de bezoeker van Docker.com de website vanaf kantoor bezoekt.

Andere interessante cijfers die we kunnen bekijken zijn die van Github, StackOverflow, Alexa en Google. Zoals in onderstaande grafiek te zien is, is de populariteit van Docker enorm toegenomen, met name in het jaar 2014.



Wanneer er gekeken wordt naar de statistieken van Github met betrekking tot Docker, dan zien we dat er (d.d. 24 december 2014) ruim 1,4K watchers zijn, bijna 18K stars zijn gegeven en ruim 3,6K forks zijn gedaan van de Docker repo. Daarnaast zijn er ruim 22K repositories die over Docker gaan. Dit in vergelijking met bijna 100K repos over NodeJS. De initiële versie van NodeJS is in mei 2009 gelanceerd. Dat betekent dat NodeJS ongeveer 4,5 jaar bestaat. Docker bestaat 1,5 jaar. Wanneer Docker met dezelfde snelheid blijft groeien als afgelopen jaar, dan bereikt Docker dezelfde populariteit als NodeJS.

Op StackOverflow.com zijn er (d.d. 24 december 2014) ruim 5,6K topics over Docker. Dit geeft aan dat Docker een grote mate van interesse heeft onder de developers. Daarnaast toont Alexa.com ons dat tussen 24 september 2014 en 24 december 2014 met ruim 6,000 plaatsen is gestegen op de wereldwijde ranglijst.

Andere bijzondere feiten over de activiteiten van de community om Docker zijn opkomende evenementen, nieuwsberichten en de bijna dagelijkse activiteiten op het forum van Docker. In de komende periode zijn er 36 evenementen over Docker gepland in Azië, Europa, Noord- en Zuid-Amerika en zelfs op eilanden in de Grote Oceaan. Verder zien we dat Docker media 2014 ertoe is overgegaan om een adviespanel op te stellen, wat duidelijk aangeeft dat er een grote interesse en sterke ontwikkeling van de Docker community plaatsvindt.

Ook de ontwikkeling van Docker staat alles behalve stil. Uit andere statistieken van Github blijkt dat er nog steeds dagelijks commits plaatsvinden in de Docker repository op Github. Een ander feit waaruit de populariteit van Docker blijkt, is dat Docker op nummer 1 staat op de lijst van open source projecten in 2014.

CONCLUSIE

Een eerste gedachte zou kunnen zijn dat, omdat Docker nog maar 'pas' ontwikkeld is, Docker nog niet klaar is om mee te doen met de grote spelers. De cijfers spreken deze gedachte echter tegen. Docker is een populair product en wordt door grote partijen als Yelp, Spotify en Ebay gebruikt. Er kan dus geconcludeerd worden dat Docker een grote community heeft.

Er wordt echter nog volop ontwikkeld en nieuwe functionaliteiten toegevoegd aan Docker. Hoewel Docker dus een grote community heeft, kan het wel nog verder ontwikkeld worden. Dit betekent echter niet dat het product te 'onvolwassen' is om te gebruiken.

Wat is de leercurve van Docker?

Het leren van Docker op zich is niet bijzonder lastig. Dat komt mede door de aanwezigheid van duidelijk documentatie. Docker beschikt over installatiehandleidingen voor onder andere Mac OS X, Ubuntu, Red Hat Enterprise Linux, Windows, CentOS en Debian. In totaal staan er 18 installatiehandleidingen op de website van Docker. Dit maakt het bijzonder eenvoudig om Docker te installeren ongeacht het besturingssysteem dat wordt gebruikt.

Daarnaast zijn er handleidingen beschikbaar voor vrijwel alle mogelijkheden die Docker biedt. Hierdoor is het relatief eenvoudig om de *ins and outs* van Docker te leren. Verder biedt de website een interactieve tutorial om aan de slag te gaan met Docker. In deze tutorial wordt je stap voor stap geholpen met het zelf opzetten van een container en het downloaden en starten van een container.

Het opzetten van een simpele container met Ubuntu 14.04, is zo simpel als onderstaand voorbeeld.

Allereerst dient er een Dockerfile gemaakt te worden, daarin kan de volgende inhoud geplaatst worden:

```
FROM ubuntu:14.04

MAINTAINER Hanze <instituut@hanze.nl>
```

Dit is de basis van een container in Docker. Door deze docker container te bouwen, wordt een installatie van Ubuntu 14.04 gedaan. Wil je op deze container ook ruby en sinatra geïnstalleerd hebben, dan zou de Dockerfile er zo uit hebben gezien:

```
FROM ubuntu:14.04

MAINTAINER Hanze <instituut@hanze.nl>

RUN apt-get update &&
    apt-get install -y ruby ruby-dev

RUN gem install sinatra
```

Op basis van deze Dockerfile kan een image gemaakt worden:

```
$ docker build -t="hanze/sinatra" .
```

Op basis van deze image kan de container gemaakt worden. De container kan vervolgens opgestart worden door het commando `docker run` te gebruiken. In onderstaand voorbeeld wordt de container opgestart en zitten we in de shell van Ubuntu:

```
$ docker run -t -i hanze/sinatra /bin/bash
```

Zoals uit bovenstaande voorbeelden duidelijk wordt, is het heel erg eenvoudig om met gebruik van Docker een omgeving op te zetten. De moeilijkheid bij Docker zit hem in de kennis die je moet hebben van het systeem dat je wilt gaan gebruiken. Echter, ook zonder het gebruik van Docker was het noodzakelijk om kennis te hebben van het systeem dat opgezet moest worden. Wanneer er een development server opgezet moet worden met Ubuntu, PHP en MySQL, dan is daar kennis van nodig. Bij het opzetten van de container moeten deze handelingen dus in de Dockerfile gezet worden, in plaats van dat deze op de server worden uitgevoerd.

Voor wat voor soort bedrijven is Docker relevant?

Dit hoofdstuk gaat wat dieper in op voor welke soort bedrijven Docker relevant is. Voordat een bedrijf het besluit neemt om gebruik te gaan maken van Docker, is het verstandig dat er bekeken wordt of de tijd die in Docker geïnvesteerd wordt, terugverdient kan worden. Aan de hand van de bedrijven die in dit hoofdstuk aangehaald worden samen met een aantal argumenten, moet het mogelijk zijn om te bepalen voor welke bedrijven Docker relevant is.

Zoals uit het hoofdstuk 'Welke bedrijven gebruiken Docker' is gebleken, zijn er veel grote bedrijven die gebruik maken van Docker. Deze bedrijven hebben aangegeven dat zij Docker gebruiken omdat daarmee alle omgevingen gelijk gehouden kunnen worden. Zij gaven daarbij aan dat ze vaak gebruik maakte van veel verschillende soorten software en libraries. Problemen worden vaak veroorzaakt door verschillen in libraries en software.

Deze grote bedrijven hebben vaak ook de benodigde kennis en expertise in huis wat betreft servermanagement. Zij hebben zelf hun servers en ontwikkelomgevingen opgezet en verzorgen ook zelf het onderhoud hiervan.

Dit in tegenstelling tot kleinere bedrijven. Er wordt hierbij wel een onderscheid gemaakt. Wanneer grote bedrijven bekeken worden, blijkt dat zij vaak een dermate grote eigen IT-afdeling hebben, dat deze afdeling bijna een bedrijf op zich genoemd kan worden. Binnen kleinere bedrijven dienen we een onderscheid te maken tussen IT gerelateerde bedrijven en bedrijven die dat niet zijn.

Kleine bedrijven die geen producten leveren op het gebied van IT, maken hier vaak wel gebruik van ter ondersteuning van hun diensten. Deze bedrijven hebben een aantal ontwikkelaars in dienst die voor hen deze ondersteuning verzorgen. Zij hebben vaak geen eigen serverbeheer, maar hebben dit bij een externe partij ondergebracht. Door het extern onder te brengen, hebben zij de servers niet zelf hoeven configureren en updaten. IT gerelateerde bedrijven doen dit, in veel gevallen, juist wel.

Zoals in andere hoofdstukken vermeld, is de leercurve van Docker op zich niet stijl (Zie het onderdeel over *“Wat is de leercurve van Docker?”*). Het is vrij eenvoudig om Docker te leren. De moeilijkheidsgraad bij het gebruik van Docker zit in het juist opzetten van de server. Grotere bedrijven, zoals bijv. Ebay, configureren zelf de servers. Kleinere niet IT gerelateerde bedrijven besteden dit vaak uit en beschikken dus niet over de benodigde expertise. Kleinere bedrijven die wel IT gerelateerd zijn, beschikken wel over deze expertise.

Deze laatste groep kleine bedrijven zal dus geen problemen ervaren bij het opzetten en configureren van de server via Docker. Zij dienen echter wel een afweging te maken. Zij dienen zich af te vragen of de investering terugverdient kan worden. Vragen die daarbij kunnen helpen zijn:

- Maken we gebruik van relatief veel verschillende soorten libraries en software?
- Ervaren we momenteel problemen bij ontwikkeling of deployment, die het gevolg zijn van verschillende versies software die gedraaid worden of libraries die ontbreken?

- Heeft ons team een dermate omvang dat het veel tijd kost om overal dezelfde versies van de software en libraries te draaien?
- Beschikken we écht over de benodigde expertises, of hebben we alleen 'wel eens wat ervaring opgedaan'?

Het beantwoorden van dergelijke vragen kan een beter inzicht bij het nemen van de beslissing of Docker wel of niet relevant is voor het bedrijf.

Hieruit kan dus de conclusie getrokken worden, dat Docker in ieder geval relevant is voor grotere bedrijven die gebruik maken van de IT. Voor kleinere bedrijven die niet beschikken over de benodigde expertise voor het opzetten en juist configureren van een server. Voor hen is het dus verstandig om geen gebruik te maken van Docker. De andere groep kleine bedrijven, kan dit het beste zelf bepalen. Voor hen is het sterk afhankelijk van de situatie waar zij zich in bevinden. De bovenstaande vragen kunnen daarbij een hulp zijn.

Welke bedrijven gebruiken Docker?

Er zijn veel verschillende bedrijven die gebruik maken van Docker, waaronder een aantal bijzonder grote bedrijven. Hieronder zullen een aantal van deze bedrijven besproken worden. Er zal eerst kort aangehaald worden wat het bedrijf is en daarna zal uitgelegd worden waar zij Docker voor gebruiken. Op deze manier kan een beeld gevormd worden van hoe belangrijk Docker inmiddels voor grote bedrijven is.

SPOTIFY

Spotify is een streamingsservice voor muziek. Zij streamen muziek naar meer dan 40 miljoen gebruikers in wereldwijd bijna 60 landen. Op de rankinglijst van Alexa is de website van Spotify in de afgelopen 3 maanden met ruim 260 plaatsen gestegen naar 391 (d.d. 24 december 2014).

Aan de back-end heeft Spotify meer dan 100 verschillende services draaien en maken ze gebruik van meer dan 5,000 servers voor de productie omgeving. Per *ops engineer* heeft Spotify zo'n 300 servers draaien.

Spotify heeft zijn continuous delivery process versnelt door gebruik te maken van Docker voor het draaien van tests en bij het deployment process. Volgens Spotify moeten deployments de volgende eigenschappen hebben:

- Ze moeten *repeatable* zijn;
- Ze moeten *straightforward* zijn;
- Ze moeten *fault-tolerant* zijn.

Omdat met Docker deze eigenschappen gewaarborgd kunnen worden, heeft Spotify gekozen om Docker te gebruiken voor het versnellen van het continuous delivery proces.

EBAY

Ebay is 's werelds grootste online marktplaats. De website van Ebay heeft op de wereldwijde ranking lijst van Alexa een plaats van 21 en op de Amerikaanse lijst een ranking van 7. Ebay gebruikt Spotify voor zijn continuous integration process voor het implementeren van een efficiënt en geautomatiseerd ontwikkeltraject.

GILT

Gilt is een leidende webshop in Amerika met meer dan 6 miljoen leden en meer dan 1,000 werknemers. Op de wereldwijde ranglijst is de website van Gilt in de afgelopen 3 maanden met bijna 1,500 plaatsen gestegen naar plaats 1,242. Op de Amerikaanse lijst staat Gilt zelfs bijna op plaats 400.

Op drukke dagen loopt het aantal productie releases van Gilt op tot zo'n 100 per dag. Door het hoge aantal releases heeft Gilt besloten om het continuous integration process te vereenvoudigen. Een kleine rekensom kan ons helpen om te begrijpen waarom. Stel dat Gilt door het vereenvoudigen van het CI-process gemiddeld per release 5 minuten zou kunnen winnen. Uitgaande van 100 releases, zoals op een drukke dag, betekent dat

een tijdwinst van ruim 8 uur per dag. Dat is (theoretisch gezien) een complete werkdag. Daarom heeft Gilt besloten om alle software naar Dockers platform te verplaatsen en zo de services te isoleren.

YELP

Yelp is een online stadsgids met actuele informatie over allerlei verschillende soorten winkels, gebaseerd op onderbouwde meningen van een community van mensen. Sommigen zullen Yelp ook kennen van Siri, een service van Apple in de vorm van een personal assistent. Yelp is een zeer populaire service, getuigd de ranking op de wereldwijde lijst van Alexa, namelijk 135.

Yelp geeft zelf aan dat Docker een vitale rol speelt in de volgende generatie van Yelps test en service infrastructuur. Yelp gebruikt Docker binnen het CI-process voor het verkorten van de ontwikkel cycli. Door het isoleren van dependencies heeft Yelp deze cycli kunnen verkorten en de tijd van het testen kunnen terugbrengen tot een kwart van wat het eerst was.

BLEACHER REPORT

Bleacher Report is onderdeel van Turner Sports en schrijft artikelen over honderden sportteams van over heel de wereld. Het is een van de snelst groeiende bedrijven in Amerika. Op de wereldwijde ranglijst van Alexa heeft Bleacher Report een ranking van 298.

Bleacher Report gebruikt Docker voor de volledige application lifecycle. Zoals zij zelf aangeven⁶⁹: "We've been very happy with the gains from using Docker and it has helped us with scale, speed, and *consistency*." Net als andere bedrijven, noemt ook Bleacher Report dat één van de voordelen van Docker consistentie is. Verder zeggen zij dat er een grote Docker-verschuiving plaatsvindt in de development wereld.

BAIDU

Baidu is dé nummer 1 Chinese internet search provider. Dat getuige ook de ranglijsten van Alexa. Op de wereldwijde lijst staat Baidu op nummer 5 en op de Chinese lijst zelfs op nummer 1.

Baidu gebruikt Docker bij de development, omdat het een aantal voordelen biedt. Zo zorgt Docker volgens hen voor een hele generieke benadering en maakt het daarnaast ook de support makkelijker voor:

- nieuwe programmeertalen;
- nieuwe frameworks;
- nieuwe databases.

Doordat er makkelijk geschakeld kan worden tussen programmeertalen, frameworks en databases kunnen tevens de kosten gereduceerd worden.

CONCLUSIE

Er zijn dus veel grote bedrijven die gebruik maken van Docker. Dit komt onder andere doordat deze grote bedrijven enorme aantallen services hebben draaien op veel verschillende servers. Door het implementeren van Docker maken zij de ontwikkeltrajecten een stabielere en betrouwbaarder.

TESTCONFIGURATIE

Deze sectie beschrijft de installatie van Docker en het opzetten van een ontwikkelomgeving. Let op dat voor dit hoofdstuk de technische kennis is vereist die kan worden verwacht van een systeembeheerder.

Installatie

Allereerst dient Docker te worden geïnstalleerd. Er is voor iedere OS een aparte handleiding⁸⁹ beschikbaar. Hieronder zal wat dieper ingegaan worden op de installatie van Docker op Mac OS X¹⁸.

Docker maakt gebruik van Linux specifieke kernel features en dient daarom via een virtuele machine (VM) gedraaid te worden. Om dit proces simpeler te laten verlopen heeft Docker applicatie ontwikkeld die Boot2docker⁹⁰ heet. Boot2docker installeert een VM die gereed is gemaakt voor het draaien van Docker.

Boot2docker kan geïnstalleerd worden via de Docker for OS X Installer⁹¹, of via Homebrew⁹². Homebrew is de package manager voor OS X. Boot2docker kan als volgt geïnstalleerd worden:

```
$ brew install boot2docker
```

Boot2docker maakt gebruik van de 'go' en 'docker' package en deze zullen dan ook automatisch worden geïnstalleerd door Homebrew. Wanneer de installatie van Boot2docker is voltooid, moet deze nog geïnitieerd worden. Dat kan als volgt:

```
$ boot2docker init
$ boot2docker up
$ $(boot2docker shellinit)
```

De eerste stap, `$ boot2docker init`, maakt een nieuwe VM. Vervolgens wordt door middel van `$ boot2docker up` de VM gestart. Het `$ $(boot2docker shellinit)` wordt gebruikt om de `DOCKER_HOST` variable te zetten in de huidige shell. Wanneer er gebruik gemaakt wordt van een andere shell zoals Fishshell⁹³, dan dient `$ $(boot2docker shellinit)` vervangen te worden⁹⁴ door `$ boot2docker shellinit | while read line; eval $line; end`.

Bij het starten van de VM wordt een feedback gegeven dat lijkt op het onderstaande:

```
$ boot2docker up
```

Vooraf de laatste is belangrijk. Wanneer de omgevingsvariabelen ('environment variables') niet goed zijn ingesteld, wordt feedback getoond die lijkt op het onderstaande:

```
To connect the Docker client to the Docker daemon, please set:

set -x DOCKER_HOST tcp://192.168.59.103:2376

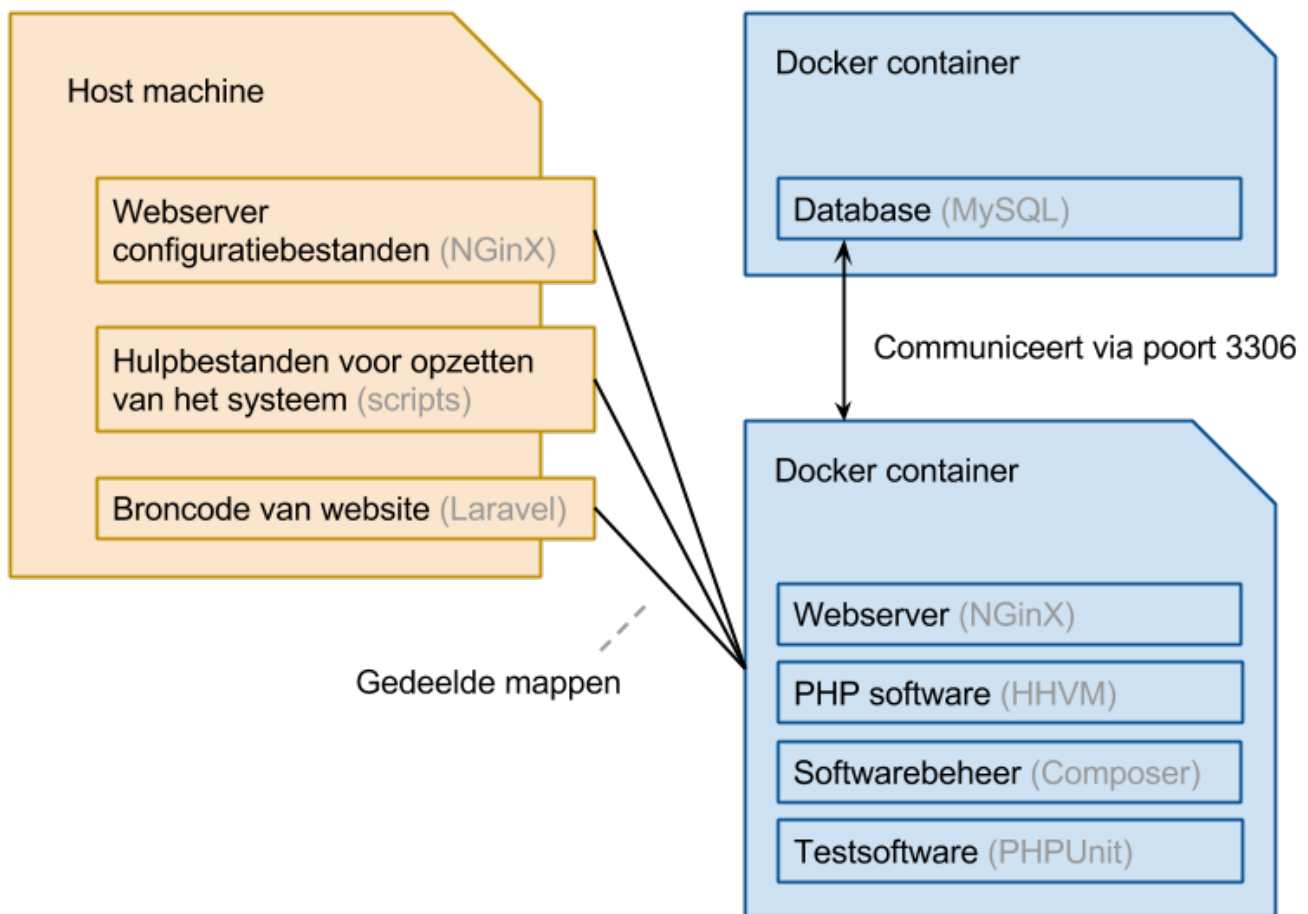
set -x DOCKER_CERT_PATH /Users/[username]/.boot2docker/certs/boot2docker-vm

set -x DOCKER_TLS_VERIFY 1
```

Dit kan verholpen worden door het `shellinit` commando te draaien. Het is verstandig om het `shellinit` commando toe te voegen aan de profile van de shell zodat niet telkens opnieuw het `shellinit` commando uitgevoerd hoeft te worden.

Opzet testomgeving

Doordat er met Docker software in geïsoleerde containers kan worden gedraaid, is het eenvoudig om een testomgeving op te zetten zonder dat dit impact heeft op de ('host') machine van de gebruiker. Een veelvoorkomende opzet is een webserver met een database. Voor de testomgeving worden dan ook de volgende softwarecomponenten (ook wel 'stack' genoemd) gebruikt:



In deze opstelling worden een groot aantal aspecten omtrent Docker gedemonstreerd:

- De Docker command line tool;
- Het gebruik van Boot2docker op een Unix systeem (Mac OSX);
- Het gebruik van bestaande images en de Docker Index;
- Een Docker image opstellen met behulp van een *Dockerfile*;
- Het installeren van software in een container;
- Het opzetten, (her)starten, stoppen, linken en inspecteren van containers;
- Het toegang verkrijgen tot containers;
- Een backup maken van de database:
 - Uitvoer van interne gegevens;
 - Het benaderen van applicaties ('services') binnen een container.

Resultaten

In deze paragraaf wordt ingegaan in de resultaten van de testconfiguratie. Zo worden de stappen toegelicht om een Docker omgeving op te zetten, met afsluitend een aantal handige tips.

STAPPEN

Voor het opzetten van de softwarecomponenten kunnen de volgende stappen worden doorlopen. Let op dat hier kennis van de command line voor is vereist en dat het is uitgevoerd op een Unix besturingssysteem (Mac OSX 10.10).

1. Start een MySQL container vanaf de bestaande image⁹⁵:

```
$ bashdocker run \
    --name "db" \
    -d -p 3306:3306 \
    -e MYSQL_ROOT_PASSWORD=banaan \
    -e MYSQL_DATABASE=laravel \
    mysql
```

2. Maak een `web/` map aan (`$ mkdir web/`) met deze bestanden die hier te vinden zijn⁹⁶. Deze *Dockerfile* installeert de volgende software:
 - NGinX⁹⁷;
 - HHVM⁹⁸, een snelle compiler voor PHP (~50% snelheidswinst);
 - Een PHP MySQL driver, om verbinding te kunnen maken met de database;
 - Composer⁹⁹, om Laravel te kunnen installeren/updaten;
 - PHPUnit¹⁰⁰, voor het testen van de PHP code;
 - Een SSH server, om de container vanaf een externe locatie te kunnen benaderen.
3. Navigeer naar de `web/` map en draai `$ docker build -t="web" .` om een 'web' image te maken van de huidige directory (`./Dockerfile`).
4. Navigeer naar de bovenliggende map (`$ cd ../`).

5. Draai het volgende commando om een Laravel site op te zetten:

```
$ docker run \  
    -it \  
    -v $(pwd)/test:/var/www/html \  
    -v $(pwd)/web/script:/var/script \  
    web \  
    /var/script/create.sh
```

6. Start een web container en bewaar de container ID:

```
$ DOCKER_WEB_ID=$(docker run \  
    --name "web" \  
    -d \  
    -p 80:80 \  
    -p 2222:22 \  
    -v $(pwd)/web/nginx:/etc/nginx/sites-enabled \  
    -v $(pwd)/web/certs:/etc/nginx/certs \  
    -v $(pwd)/web/logs:/var/log/nginx \  
    -v $(pwd)/web/script:/var/script \  
    -v $(pwd)/site:/var/www/html \  
    --link db:db \  
    web)
```

Let op: In Windows is het resultaat van een commando opslaan (middels `$ (..commando..)`) in een variabele (`VARIABLE=$ (..commando..)`) niet mogelijk. Vervang daarom in de volgende stappen `$DOCKER_WEB_ID` met de uitvoer van het, in deze stap, uitgevoerde commando. Vervang op Windows ook `$ (pwd)` met `%cd%`.

7. Verbind met de web container om de **NGinX** log te zien:

```
$ docker attach --sig-proxy=false $DOCKER_WEB_ID
```

8. Tot slot moet het IP adres van de server worden achterhaald om de website te openen. Bij een ander besturingssysteem dan Linux wordt er gebruik gemaakt van *Boot2docker*. Dit houdt in dat het IP adres van de virtuele machine moet worden gebruikt in plaats het IP adres van de container zelf. Dit kan worden gedaan met `$ boot2docker ip` (het adres van de container kan worden achterhaald met `$ docker inspect --format '{{.NetworkSettings.IPAddress}}' "$DOCKER_WEB_ID"`).
9. Open tot slot het IP adres in een webbrowser om het resultaat te bekijken.

OVERIG

Dit voorbeeld geeft een inzicht in het opzetten van een ontwikkelomgeving met behulp van Docker. Afsluitend nog een aantal tips:

- **Layer caching:** Bij het omzetten van een *Dockerfile* naar een image worden alle 'RUN' commando's in aparte containers uitgevoerd. Docker cached¹⁰¹ op deze manier alle stappen van het 'build process', om bij het opnieuw bouwen van de image aanzienlijke snelheidswinst te behalen.
- **Herstarten container:** Met het `$ docker start <container ID>` commando kan een gestopte container opnieuw worden gestart.
- **Toegang tot container:** Er kan een andere versie van de container worden opgezet of een SSH server worden geïnstalleerd:
 1. De start opdracht van een container kan worden overschreven met het `$ docker -it run <image> /bin/bash` commando. Hiermee wordt er Bash¹⁰² gestart om toegang te krijgen tot de command line van de container zelf.

Om toegang te krijgen tot de web container (zie ook stap 6) kan het volgende commando worden gedraaid:

```
$ DOCKER_WEB_ID=$(docker run \
    -it \ # Command line modus
    -v $(pwd)/web/nginx:/etc/nginx/sites-enabled \
    -v $(pwd)/web/certs:/etc/nginx/certs \
    -v $(pwd)/web/logs:/var/log/nginx \
    -v $(pwd)/web/script:/var/script \
    -v $(pwd)/site:/var/www/html \
    --link db:db \
    web /bin/bash) # Run Bash
```

Let op: De regels die met `-v` beginnen maken mappen van de host machine beschikbaar aan de container. Deze zijn optioneel.

2. Alternatief kan er een SSH-server¹⁰³ worden geïnstalleerd middels de *Dockerfile* om zo SSH¹⁰⁴ toegang te krijgen tot de container. Deze methode heeft als voordeel dat er eenvoudig kan worden geïnspecteerd wat een container doet die wel de start opdracht draait.

In het voorgaande voorbeeld wordt een SSH-server geïnstalleerd, dus kan er met het volgende commando toegang worden verkregen tot de draaiende web container:

```
$ ssh -p 2222 root@$ (boot2docker ip)
```

3. Sinds oktober 2014 maakt Docker het mogelijk om direct commando's uit te voeren in draaiende containers. Dit kan worden gedaan met het `exec` commando:

```
$ docker exec -it web /bin/bash # Voor een interactieve Bash shell
```


- **MySQL data backup:** Om een backup te maken van de data in de database container kan er `$ docker exec db mysqldump --all-databases --password=banaan` worden gedraaid.

Bevindingen

Tijdens het uitvoeren van de test met Docker, zijn we tegen een aantal dingen aangelopen. Het installeren van Docker en het gebruik van bestaande softwarecomponenten bleek vrij eenvoudig te zijn. De obstakels begonnen zich daarna pas voor te doen. Hieronder zullen we er iets dieper op ingaan.

OPZETTEN VAN DE CONTAINER

In de praktijk zijn servers vaak op maat geconfigureerd. Tijdens dit onderzoek wilden wij dus ook een server opzetten aan de hand van een zelf samengestelde Dockerfile, in plaats van overal bestaande softwarecomponenten voor te gebruiken. Hierbij hebben we dus zelf na moeten denken over vragen als 'Welk OS gaan we gebruiken?', 'Welke versie van PHP gaan we draaien?', 'Gaan we wel of geen HipHop Virtual Machine gebruiken?' en 'Draaien we de database in een aparte container of dezelfde container?'.

Daarnaast moesten we deze componenten dus ook zelf installeren in onze container. De eerste twee obstakels die we tegenkwamen betroffen dus de kennis die benodigd was voor zowel het maken van de juiste keuzes, als voor de installatie van de verschillende componenten.

IP-ADRES VAN DOCKER

De server die binnen de container van Docker draait, kan niet benaderd worden via de standaard localhost. Daartoe dient het IP-adres van de container zelf gebruikt te worden. Echter, bij een Mac werkt het anders. Docker kan alleen op een Mac draaien door gebruik te maken van een VM. Hiervoor kan boot2docker gebruikt worden. Dit heeft tot gevolg dat, bij het benaderen van de server, niet het IP-adres van de container moet worden gebruikt, maar het IP-adres van boot2docker.

NGINX EN HHVM

Het bleek lastig om NGinX en HHVM tegelijk te laten draaien. Het lukte niet om de services als daemon te laten draaien via de container. Ergens voor het einde van het installatieproces werden de services dan weer afgesloten. Wanneer NGinX niet als daemon werd gedraaid, bleef de rest van de scripts na het starten van NGinX te worden geblokkeerd. Hiertoe hebben we een shell script gemaakt dat beide processen als daemon kan starten. Het gebruikte script staat hieronder:

```
# Start services

/usr/sbin/sshd -D &

service hhvm start

service nginx start
```

KOPPELEN VAN COMPONENTEN

Het voordeel van Docker is dat container gelinkt kunnen worden. Op die manier kan de container die de database bevat los van de container met de container met de server worden gedraaid. Het was niet mogelijk om de environmentvariabelen te benaderen, wat het lastig maakte om bepaalde data door te geven.

GROTE MYSQL CONTAINER

Voor onze server hebben we gebruik gemaakt van de originele MySQL image. Echter, na installatie van de MySQL-server wordt de broncode niet verwijderd. Dit resulteert in een image van 2,5 GB.

CONCLUSIE

Zoals uit het hoofdstuk *“Wat is de leercurve van Docker?”* blijkt, is het opzetten van Docker in de basis vrij eenvoudig. Het opzetten van de basis van Docker is echter niet de enige stap die dient te worden uitgevoerd. Vaak zijn servers op maat geconfigureerd. Dit heeft tot gevolg dat er ook enige kennis is vereist voor het opzetten en configureren van een server.

Uit de hoofdstukken *“Welke bedrijven gebruiken Docker?”* en *“Voor wat voor soort bedrijven is Docker relevant?”* blijkt dat er een onderscheid gemaakt kan worden tussen de verschillende soorten bedrijven die gebruik maken van Docker. Er zijn grofweg drie soorten bedrijven: (1) grote bedrijven, (2) kleinere, **niet** IT gerelateerde, bedrijven en (3) kleinere, **wel** IT gerelateerde, bedrijven.

Grote bedrijven draaien vaak grote aantallen verschillende diensten ('services') op verschillende servers. Voor hen is, de door Solomon Hykes gebruikte, term 'Matrix of Hell' goed van toepassing. Dit zijn de verschillende combinaties in hardware en softwareversies die ongewenste problemen veroorzaken en het werk van systeembeheerders een stuk complexer maakt. Deze grote bedrijven beschikken vaak wel over de benodigde investering van tijd en expertise voor het opzetten en juist configureren van een server. Door het implementeren van Docker maken zij de ontwikkeltrajecten stabiel en betrouwbaarder.

Binnen de kleinere bedrijven kan onderscheid worden gemaakt tussen de bedrijven die IT diensten leveren en de bedrijven die dat niet doen. Kleine bedrijven die geen IT diensten leveren, hebben het beheer van de server in de meeste gevallen uitbesteedt. Daardoor beschikken zij niet over de benodigde expertise. Voor hen is het dus verstandig om niet over te stappen op Docker of zich hierover in te laten lichten door experts.

De kleinere bedrijven die wel IT diensten leveren, kunnen deze afweging het beste zelf maken. Voor hen is het sterk afhankelijk van de situatie waar zij zich in bevinden. In het hoofdstuk over *“Voor welke soort bedrijven is Docker relevant?”* staan een aantal vragen die hierbij kunnen helpen. Zij dienen tevens rekening te houden met het feit dat Docker in principe nog in de kinderschoenen staat. Hoewel er op StackOverflow.com (d.d. 24 december) ruim vijfduizend topics over Docker staan en de community rondom Docker nog in de kinderschoenen staat, vindt er ook nog steeds veel ontwikkeling plaats aan Docker zelf. Dit betekent dat er nog genoeg functionaliteiten aan Docker ontbreken. Grotere bedrijven die over genoeg kennis beschikken, kunnen deze gaten vaak makkelijker opvangen dan kleinere bedrijven die wellicht minder expertise in huis hebben.

Een ander belangrijk aspect dat in overweging genomen dient te worden, is de complexiteit van de applicatie architectuur. Wanneer deze uit veel nauw samenwerkende componenten bestaat, wordt het opzetten en beheren van deze componenten erg complex en is Docker niet altijd de juiste oplossing. Zo heeft het weinig nut om de gehele applicatie in een enkele container te draaien, terwijl de software ook op de daadwerkelijke server gedraaid zelf kan worden. In dergelijke gevallen kan de isolatie soms meer tijd en moeite kosten dan dat het uiteindelijk oplevert. En zo biedt de overhead van Docker mogelijk geen meerwaarde bij applicaties waarin falende componenten een lage impact hebben.

DISCUSSIE

Door het korte onderzoekstraject is er enkel online literatuur gebruikt die op het moment van schrijven aanwezig was. Het onderzoek heeft daarom een beperkt gezichtsveld, omdat er geen onderzoek is gedaan naar onder andere de directe bevindingen van (Nederlandse) Docker gebruikers. Statistieken van Docker gebruik in het MKB van Nederland kan wellicht inzicht geven in het soort bedrijf dat Docker gebruikt.

Additioneel dient er de stabiliteit van (langer) draaiende containers onderzocht te worden. Het is uiteraard mogelijk om containers opnieuw te starten bij falen, echter is symptoombestrijding, terwijl een dergelijk probleem bij de bron aangepakt dient te worden.

Een vervolgstap voor deze literatuurstudie zou zijn om de gedane bevindingen te presenteren in de vorm van een checklist. Dit kan onervaren bedrijven/personen op een eenvoudige manier inzicht geven of Docker meerwaarde biedt voor hun huidige of gewenste situatie.

In het kader van het onderzoek kan er tevens een testcase worden opgezet met bedrijven die Docker gebruiken of willen gebruiken. Deze partijen kunnen worden gemonitord om meer inzicht te verkrijgen in problemen, oplossingen en voor- en nadelen in praktijksituaties.

Tot slot kan de veiligheid van Docker (containers) verder worden onderzocht. Hoe groot zijn de huidige veiligheidsproblemen? Zijn deze inmiddels opgelost? Wat zijn scenario's waar de veiligheid niet kan worden gewaarborgd? Goede resultaten op dit gebied kunnen doorslaggevend zijn in de keuze.

BRONNEN

1. <https://www.youtube.com/watch?v=wW9CAH9nSLs>
2. <https://www.peperzaken.nl/blog/docker-cutting-edge-code-shipping>
3. <http://techcrunch.com/2014/01/21/docker-raises-15m-for-popular-open-source-platform-designed-for-developers-to-build-apps-in-the-cloud/>
4. Leertaak-3-Onderzoek.pdf (van blackboard)
5. <http://uk.businessinsider.com/docker-a-hugely-important-startup-2014-11>
6. <https://gigaom.com/2014/06/10/why-companies-like-google-spotify-and-red-hat-are-embracing-dockers-open-source-containers/>
7. <http://blog.docker.com/category/docker-releases/>
8. <http://www.wired.com/2014/11/following-google-microsoft-amazon-embraces-next-big-thing-cloud-computing/>
9. <https://github.com/googlecloudplatform/kubernetes>, GoogleCloudPlatform
10. <http://stackoverflow.com/questions/16047306/how-is-docker-io-different-from-a-normal-virtual-machine>
11. <http://www.techrepublic.com/article/why-docker-and-why-now/>
12. <https://www.youtube.com/watch?v=wW9CAH9nSLs>
13. <http://en.wikipedia.org/wiki/UnionFS>
14. <https://www.docker.com/whatisdocker/>
15. http://blog.docker.com/wp-content/uploads/2013/08/the_matrix_of_hell.jpg
16. <https://docs.docker.com/introduction/understanding-docker/>
17. <https://linuxcontainers.org>
18. <https://docs.docker.com/installation/mac/>
19. <http://www.projectatomic.io/blog/2014/08/is-it-safe-a-look-at-docker-and-security-from-linuxcon/>
20. <https://titanous.com/posts/docker-insecurity>
21. <http://www.vmware.com/>
22. <https://www.virtualbox.org/>
23. <https://www.vagrantup.com/>
24. <http://puppetlabs.com/>

25. <https://www.chef.io/>
26. <http://www.ansible.com/>
27. <http://www.flockport.com/>
28. <https://coreos.com/blog/rocket/>
29. <https://linuxcontainers.org/lxd/introduction/>
30. <http://www.slideshare.net/dotCloud/why-docker>
31. <http://www.slideshare.net/dotCloud/why-docker>
32. <http://www.quora.com/What-are-the-pros-and-cons-of-running-one-process-per-Docker-container>
33. <http://stackoverflow.com/questions/16047306/how-is-docker-io-different-from-a-normal-virtual-machine>
34. <http://sathishmanohar.com/articles/docker-will-change-everything/>
35. <http://sysadvent.blogspot.nl/2014/12/day-1-docker-in-production-reality-not.html>
36. <http://docs.docker.com/introduction/technology/>
37. <http://kubernetes.io/>
38. <https://github.com/zettio/weave>
39. <https://github.com/coreos/flannel>
40. <https://github.com/jpetazzo/pipework>
41. <http://socketplane.io/>
42. <http://mesos.apache.org/>
43. <http://thenewstack.io/how-docker-fits-into-the-devops-ecosystem/>
44. <http://www.informationweek.com/cloud/infrastructure-as-a-service/chef-finds-docker-a-close-fit/d/d-id/1297282>
45. <http://recursivity.com/blog/2014/09/08/how-ansible-docker-fit-using-ansible-to-bootstrap-coordinate-docker-containers/>
46. <http://www.wired.com/2014/06/eric-brewer-google-docker/>
47. <http://computerworld.nl/beveiliging/84279-5-projecten-die-docker-completeren>
48. <http://www.google.com/trends/explore#q=%2Fm%2F0wkcjgj&date=1%2F2013%2024m&cmpt=q>
49. <https://www.docker.com/company/aboutus/>
50. <http://en.wikipedia.org/wiki/Node.js>
51. <https://github.com/search?utf8=%E2%9C%93&q=node>

52. <https://github.com/docker/docker>
53. <https://github.com/search?utf8=%E2%9C%93&q=docker>
54. <https://github.com/search?utf8=%E2%9C%93&q=docker+created%3A%3E2014-01-01&type=Repositories&ref=advsearch&l=>
55. <https://github.com/search?utf8=%E2%9C%93&q=docker+created%3A%3E2014-06-01&type=Repositories&ref=advsearch&l=>
56. <http://stackoverflow.com/search?q=docker>
57. <http://www.alexa.com/siteinfo/docker.com>
58. <https://www.docker.com/community/events/>
59. <https://forums.docker.com/top/daily>
60. <https://www.docker.com/company/news/>
61. <https://www.docker.com/community/governance/>
62. <https://github.com/docker/docker/graphs/commit-activity>
63. <http://www.alexa.com/siteinfo/docker.com#demographics>
64. <http://opensource.com/business/14/12/top-10-open-source-projects-2014>
65. <https://www.docker.com/tryit/>
66. <https://docs.docker.com/installation/#installation>
67. <https://docs.docker.com/installation/#user-guide>
68. <https://docs.docker.com/userguide/dockerimages/#building-an-image-from-a-dockerfile>
69. <https://www.docker.com/resources/usecases/>
70. <https://blog.docker.com/2014/07/dockercon-video-immutable-infrastructure-with-docker-and-ec2/>
71. <https://blog.docker.com/2014/07/dockercon-video-building-a-smarter-application-stack/>
72. <http://sauceio.com/index.php/2014/12/ci-cd-with-docker-beanstalk-circleci-slack-gantree/>
73. <https://blog.docker.com/2014/06/dockercon-video-docket-at-spotify-by-rohan-singh/>
74. <http://blog.docker.com/2013/12/baidu-using-docker-for-its-paas/>
75. <https://speakerdeck.com/teddziuba/docker-at-ebay>
76. http://www.yelp.nl/faq#what_is_yelp
77. <http://bleacherreport.com/about>
78. http://en.wikipedia.org/wiki/Bleacher_Report

79. <http://www.alex.com/siteinfo/bleacherreport.com#rank-panel>
80. <http://www.alex.com/siteinfo/gilt.com#rank-panel>
81. <http://www.alex.com/siteinfo/yelp.com#rank-panel>
82. <http://www.alex.com/siteinfo/spotify.com#rank-panel>
83. <http://www.alex.com/siteinfo/baidu.com#rank-panel>
84. <http://www.alex.com/siteinfo/ebay.com#rank-panel>
85. [http://www.ebayinc.com/who we are/one company](http://www.ebayinc.com/who_we_are/one_company)
86. <http://en.wikipedia.org/wiki/Spotify>
87. [http://en.wikipedia.org/wiki/Gilt Groupe](http://en.wikipedia.org/wiki/Gilt_Groupe)
88. [http://en.wikipedia.org/wiki/Siri#Research and development](http://en.wikipedia.org/wiki/Siri#Research_and_development)
89. <https://docs.docker.com/installation/#installation>
90. <https://github.com/boot2docker/boot2docker>
91. <https://github.com/boot2docker/osx-installer/releases/latest>
92. <http://brew.sh/>
93. <http://fishshell.com/>
94. <http://stackoverflow.com/a/27529061/1453912>
95. https://registry.hub.docker.com/_/mysql/
96. <https://github.com/MalcolmK/thema-4.2-se-leertaken/tree/master/onderzoek/demo/web>
97. <http://nginx.org/en/>
98. <http://hhvm.com/>
99. <https://getcomposer.org/>
100. <https://phpunit.de/>
101. [https://docs.docker.com/articles/dockerfile best-practices/#build-cache](https://docs.docker.com/articles/dockerfile_best-practices/#build-cache)
102. <http://www.gnu.org/software/bash/>
103. [http://en.wikipedia.org/wiki/Comparison of SSH servers](http://en.wikipedia.org/wiki/Comparison_of_SSH_servers)
104. [http://en.wikipedia.org/wiki/Secure Shell](http://en.wikipedia.org/wiki/Secure_Shell)