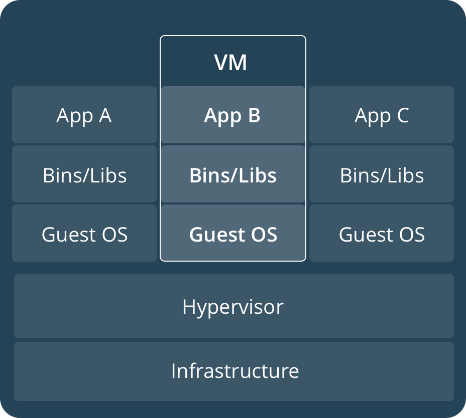
# Over containers en virtual machines

Containers en virtual machines zijn twee verschillende technologieën, maar ze kunnen beide gebruikt worden voor een gelijkaardig doel: het isoleren van een applicatie en de betreffende dependencies in een self-containede eenheid.

Verder verminderen VM’s en containers de nood aan fysieke hardware door efficiëntere benutting van computing resources.

Nu, wat is het verschil tussen virtual machines en containers, en meer bepaald, wat zijn de gevolgen hiervan?

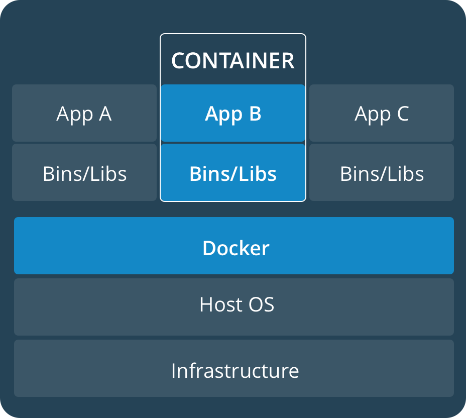
## Virtual machines

In essentie is een virtual machine een soort emulatie van een volwaardige “computer” dat een operating system runt en programma’s uitvoert alsof het een echte computer zou zijn. VM’s lopen op een fysieke machine door middel van een hypervisor. En een hypervisor draait ofwel op een host machine (hosted hypervisor) of rechtstreeks op de hardware (native of bare-metal hypervisor).

Een hypervisor zelf runt dus op een fysiek systeem, ook wel bekend als “host machine”. De host machine voorziet de VM’s van resources als RAM, en CPU. Deze resources kunnen worden verdeeld zodat ze best aansluiten aan uw use-case.

Een belangrijke eigenschap van VM’s is dat ze een volledige gevirtualiseerde hardware stack met zich mee brengen. Dit houdt dingen in als CPU, RAM, opslag, netwerk adapters, en meer. Verder heeft het ook zijn eigen volwaardig operating system.

## Container

Het een belangrijk verschil tussen een container en een VM is dat een container geen hardware virtualisatie doet, en bijgevolg niet zijn eigen operating system draait, maar in plaats daarvan wel een user space isoleert.

Een container heeft wel het een en ander gemeen met een VM, namelijk: de mogelijkheid om een filesystem te mounten, een geïsoleerde ruimte om processen in te runnen en kunnen commands uitvoeren als root binnen hun user space.

Het belangrijkste verschil is waarschijnlijk dat containers de host system kernel delen met de andere containers die draaien op de host machine.

UPDATE:(<http://www.floydhilton.com/docker/2017/03/31/Docker-ContainerHost-vs-ContainerOS-Linux-Windows.html>)

Dit is het geval bij containers op linux op windows zonder hyper-V. Wanneer je in windows hyper-V gebruikt zit ge met een extra virtualisatielaag.

## Docker

Docker is een platform gebaseerd op Linux containers. Het gebruikt Linux kernel features zoals namespaces en control groups om containers te maken bovenop en operating system.

Men gebruikt docker om applicaties te ontwikkelen, deployen en runnen door middel van containers.

### Docker Image

Een container wordt gelanceerd door een image te runnen. Een image is een executable package dat alles omvat dat je nodig hebt om een toepassing te kunnen runen; de code, een runtime, libraries, omgevingsvariabelen, en configuratie bestanden. Een lopende container is dus wat een image wordt als deze wordt uitgevoerd.

Een image definieert hoe de uiteindelijke toepassing eruit gaat zien en welke processen er gerunt worden als de container word gelaunched.

### Docker engine

De Docker engine is een lichte runtime en verzameling van tools dat containers, images, builds, en meer beheert. Het kan native op Linux runnen en bestaat voornamelijk uit een Docker Daemon en een Docker Client.

De docker client is in essentie de UI van Docker, het is hetgeen dat tussen de gebruiker en de daemon staat. De Docker daemon is wat de instructies effectief gaat uitvoeren, zoals builden, runnen, etc. De Docker daemon moet steeds op het host OS draaien.

### Dockerfile

De Dockerfile is waar je de instructies schrijft die docker interpreteert om een image mee te builden. Eens je de Dockerfile hebt geschreven kan je de docker build command gebruiken om de image uit de file te builden.

### Volumes

Een volume is het data gedeelte van een container, geïnitialiseerd wanneer de container wordt gestart. Volumes staan apart van het default Union filesystem en bestaan als gewone mappen en files in het host filesystem. Dit laat toe data bij te houden als de container wordt gesloten, verwijdert of gerebuild. Verder kan er op deze manier data uitgewisseld worden met het host filesystem en eventueel andere containers.

### Unionfs

Awel, da snap ik eigenlijk ni helemaal.

Houdt dat in dat gemeenschappelijke files gedeeld worden door containers die gebuild zijn van dezelfde image, of sla ik de bal hier helemaal mis? En wat is da met die uitleg da die writable lijken maar da ni zijn? Leg mij da nekeer uit

Nog iets tof is dat docker een netwerkinterface voorziet in de container zodat deze zo kan communiceren met de localhost en een ip krijgt.

<https://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b>