Les 1 – Docker (DevOps) – Uitgebreide Cursus

1. Introductie & Motivatie

DevOps en Docker

- DevOps = samenwerking tussen development en operations.
- Doel: snellere en betrouwbaardere softwareontwikkeling en releases.
- Containerisatie met Docker is een essentieel hulpmiddel in DevOps.

Probleemstelling

- Klassiek probleem: *“It works on my machine!”*
- Applicaties draaien lokaal maar falen in test of productie.
- Oorzaken: verschillen in OS, libraries, configuratie.

Oplossing

- Containers zorgen voor een uniforme, reproduceerbare omgeving.
- Applicaties werken overal hetzelfde, ongeacht infrastructuur.

2. Docker Fundamentals

Wat is Docker?

- Docker is een platform voor **containerisatie**.
- **Container**: standaard eenheid van software met code, afhankelijkheden en runtime.
- **Image**: het blueprint (sjabloon) waaruit containers worden gestart.
- **Docker Engine**: de runtime die containers uitvoert en beheert.
- **Docker Hub**: centrale repository om images te delen.

Containers vs Virtuele Machines

Containers
- Delen de kernel van het host-besturingssysteem.
- Lichtgewicht, typisch enkele MB’s.
- Starten in seconden.
- Minder overhead.

Virtuele Machines (VM’s)
- Emuleren volledige hardware en besturingssysteem.
- Groot (GB’s).
- Trager op te starten.
- Sterkere isolatie, maar hogere kost.

Samengevat

- Containers = lichtgewicht, snel, efficiënt en portable.
- VM’s = zwaarder, trager, maar volledig geïsoleerd.

3. Basiscommando’s en Opties

`docker run` (container starten)

Belangrijkste opties:
- `-it`: interactive terminal (gebruiksvriendelijk bij bash).
- `--rm`: verwijder container automatisch zodra hij stopt.
- `--name mycontainer`: geef container een duidelijke naam.
- `-d`: run in **detached mode** (achtergrond).
- `-p 8080:80`: map poort **8080 van host** naar **80 in container**.
- `-v /host/path:/container/path`: volume (host ↔ container).

Voorbeelden

- Een tijdelijke Ubuntu container starten en direct verwijderen na gebruik:
docker run -it --rm ubuntu bash
- Een container met vaste naam starten:
docker run -it --name mijncontainer ubuntu bash

Containers beheren

- Lijst actieve containers:
docker ps
- Lijst alle containers (ook gestopt):
docker ps -a
- Stop container:
docker stop
- Verwijder container:
docker rm

Inspectie

- Gedetailleerde info over een container of image:
docker inspect
Toont JSON met: IP-adres, mounts, netwerkconfiguratie, gebruikte image, etc.

4. Data en Volumes

Data in containers

- Bestanden in containers bestaan zolang de container leeft.
- Nieuwe container starten = schone slate (geen data van vorige container).

Types volumes

1. **Ephemeral (anonieme) volumes**
- Worden aangemaakt zonder naam.
- Verdwijnen automatisch als de container verwijderd wordt.

2. **Named volumes**
- Persistente opslag beheerd door Docker.
- Voorbeeld:
docker volume create mijnvolume
docker run -v mijnvolume:/data ubuntu

3. **Bind mounts**
- Mappen van de host koppelen aan container.
- Voorbeeld:
docker run -v /home/user/data:/app/data ubuntu
- **Links van :** = host path, **rechts van :** = container path.

Belangrijke aandachtspunten

- Bij verwijderen container:
- Data in bind mounts blijft bestaan (want op host).
- Ephemeral volumes verdwijnen.
- Named volumes blijven bestaan tenzij expliciet verwijderd (`docker volume rm`).
- Bij bind mounts kunnen **conflicten** ontstaan:
- Bestanden op host overschrijven bestanden in container.
- Container ziet altijd de versie van de host.

5. Networking en Poorten

Standaard netwerk

- Containers draaien standaard in een **bridge netwerk**.
- Containers binnen dit netwerk kunnen elkaar bereiken via container name.

Poorten mappen

- `-p hostpoort:containerpoort`
- Links: poortnummer van **host**.
- Rechts: poortnummer binnen de **container**.

Voorbeeld:
docker run -d -p 8080:80 nginx
- `8080` = hostpoort → toegankelijk via `http://localhost:8080`.
- `80` = containerpoort waar nginx luistert.

Communicatie tussen containers

- Containers kunnen elkaar pingen binnen hetzelfde netwerk.

- Voorbeeld:
docker network create mijnnet
docker run -dit --name c1 --network mijnnet ubuntu
docker run -dit --name c2 --network mijnnet ubuntu
docker exec -it c1 ping c2

Demo met netcat

- Container 1 luistert op poort 1234:
docker run -it --rm --name server ubuntu bash
apt update && apt install -y netcat
nc -lp 1234
- Container 2 stuurt bericht:
docker run -it --rm --network container:server ubuntu bash
apt update && apt install -y netcat
echo "Hallo van c2" | nc localhost 1234

6. Eigen Images met `docker commit`

Container aanpassen en opslaan

- Start een container en maak wijzigingen, bv. een bestand toevoegen:
docker run -it ubuntu bash
echo "Hallo Docker" > /hallo.txt
exit

Nieuwe image maken via commit

- Sla de gewijzigde container op als nieuwe image:
docker commit mijnimage:v1

Controleren

- Lijst images bekijken:
docker images
- Nieuwe container starten van de image:
docker run -it mijnimage:v1 bash
cat /hallo.txt

7. Publiceren naar Docker Hub

Aanmelden

docker login

Image taggen

docker tag mijnimage:v1 gebruikersnaam/mijnimage:v1

Pushen naar Docker Hub

docker push gebruikersnaam/mijnimage:v1

Controleren

- Bekijk je image op https://hub.docker.com.

8. Practica (Labs)

1. Installeer Docker op je host of VM.
2. Run 2 Ubuntu containers en ping elkaar.
3. Run 2 containers en verstuur berichten via netcat (direct & via poort mapping).
4. Maak een bestand in een container, gebruik `docker commit` en controleer of het bestand bewaard blijft.
5. Run een container met een **volume** en schrijf data naar de host. Controleer na verwijderen van de container.
6. Voeg een derde container toe als relay voor berichten (met netcat).
7. Start een container en voeg een extra proces toe vanuit een andere terminal (bv. `top`).
8. Zoek een image op Docker Hub en run deze lokaal.

9. Samenvatting

- Docker = standaard voor containerisatie.
- Containers zijn lichtgewicht, snel, portable en veilig.
- Belangrijkste opties bij `docker run`: `-it`, `--rm`, `--name`, `-d`, `-p`, `-v`.
- Dataopslag kan via ephemeral volumes, named volumes of bind mounts.
- Networking: standaard bridge, poortmapping en container-naar-container communicatie.
- `docker commit` laat je aangepaste images opslaan.
- Images kunnen gepubliceerd worden naar Docker Hub.
- Volgende les: **Dockerfile** en **Docker Compose**.