Systems Advanced Docker Containers

Container Networking

DE HOGESCHOOL MET HET NETWERK

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be

Module opbouw

In het eerste deel van de module gaan we onder de motorkap kijken van docker networking. Hiervoor gebruiken we een **ubuntu VM**, zodat we kunnen ontdekken hoe docker networking is geïmplementeerd met de linux networking stack.

Tijdens het tweede deel behandelen we hoe docker networks gebruikt kunnen worden in containers. Hiervoor kan je zowel de Ubuntu VM als Docker Desktop for Windows gebruiken.

Docker networking: onder de motorkap

Docker networking wordt verzorgd door de **`docker0`** bridge.

- `ip a`
  - We zien een **`docker0`** interface. Dat is eigenlijk een bridge of virtuele switch die softwarematig is aangemaakt door de kernel.
- `sudo apt install bridge-utils`
- `brctl show docker0`
  - We zien bij de interfaces misschien nog niets, want hier zien we de interfaces van de draaiende containers.

```
student@ubuntu-server-2:~$ brctl show docker0
bridge name     bridge id               STP enabled     interfaces
docker0         8000.0242297eb9c7       no
student@ubuntu-server-2:~$
```

BusyBox container

We gebruiken de **BusyBox** container om networking mee te testen.

BusyBox combineert kleine versies van veel voorkomende UNIX-programma's in een enkel klein programma. Het biedt minimalistische vervangingen voor de meeste programma's die je gewoonlijk in GNU coreutils, util-linux, etc. vindt. De programma's in BusyBox hebben over het algemeen minder opties dan hun volledige GNU neven; maar de opties die zijn opgenomen bieden de verwachte functionaliteit en gedragen zich heel erg als hun GNU tegenhangers.

De BusyBox container wordt door Docker zelf onderhouden. Typisch wordt deze container gebruikt voor networking debugging op een host OS, of networking debugging tussen containers of in Kubernetes.

Op je host os kan je gewoon **`docker run busybox <command>`** gebruiken. Je hebt dan alle commando's ter beschikking van BusyBox zonder die te hoeven installeren.

```
# thraa @ DESKTOP-TOMC in ~ [15:49:15]
$ docker run busybox ping -c 3 google.com
PING google.com (142.251.36.46): 56 data bytes
64 bytes from 142.251.36.46: seq=0 ttl=37 time=18.151 ms
64 bytes from 142.251.36.46: seq=1 ttl=37 time=20.431 ms
64 bytes from 142.251.36.46: seq=2 ttl=37 time=14.966 ms

--- google.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 14.966/17.849/20.431 ms

# thraa @ DESKTOP-TOMC in ~ [15:49:25]
$ _
```

Currently available applets include:

[, [[, acpid, addgroup, adduser, adjtimex, ar, arp, arping, ash, awk, basename, beep, blkid, brctl, bunzip2, bzcat, bzip2, cal, cat, catv, chat, chattr, chgrp, chmod, chown, chpasswd, chpst, chroot, chrt, chvt, cksum, clear, cmp, comm, cp, cpio, crond, crontab, cryptpw, cut, date, dc, dd, deallocvt, delgroup, deluser, depmod, devmem, df, dhcprelay, diff, dirname, dmesg, dnsd, dnsdomainname, dos2unix, dpkg, du, dumpkmap, dumpleases, echo, ed, egrep, eject, env, envdir, envuidgid, expand, expr, fakeidentd, false, fbset, fbsplash, fdflush, fdformat, fdisk, fgrep, find, findfs, flash_lock, flash_unlock, fold, free, freeramdisk, fsck, fsck.minix, fsync, ftpd, ftpget, ftpput, fuser, getopt, getty, grep, gunzip, gzip, hd, hdparm, head, hexdump, hostid, hostname, httpd, hush, hwclock, id, ifconfig, ifdown, ifenslave, ifplugd, ifup, inetd, init, inotifyd, insmod, install, ionice, ip, ipaddr, ipcalc, ipcrm, ipcs, iplink, iproute, iprule, iptunnel, kbd_mode, kill, killall, killall5, klogd, last, length, less, linux32, linux64, linuxrc, ln, loadfont, loadkmap, logger, login, logname, logread, losetup, lpd, lpq, lpr, ls, lsattr, lsmod, lzmacat, lzop, lzopcat, makemime, man, md5sum, mdev, mesg, microcom, mkdir, mkdosfs, mkfifo, mkfs.minix, mkfs.vfat, mknod, mkpasswd, mkswap, mktemp, modprobe, more, mount, mountpoint, mt, mv, nameif, nc, netstat, nice, nmeter, nohup, nslookup, od, openvt, passwd, patch, pgrep, pidof, ping, ping6, pipe_progress, pivot_root, pkill, popmaildir, printenv, printf, ps, pscan, pwd, raidautorun, rdate, rdev, readlink, readprofile, realpath, reformime, renice, reset, resize, rm, rmdir, rmmod, route, rpm, rpm2cpio, rtcwake, run-parts, runlevel, runsv, runsvdir, rx, script, scriptreplay, sed, sendmail, seq, setarch, setconsole, setfont, setkeycodes, setlogcons, setsid, setuidgid, sh, sha1sum, sha256sum, sha512sum, showkey, slattach, sleep, softlimit, sort, split, start-stop-daemon, stat, strings, stty, su, sulogin, sum, sv, svlogd, swapoff, swapon, switch_root, sync, sysctl, syslogd, tac, tail, tar, taskset, tcpsvd, tee, telnet, telnetd, test, tftp, tftpd, time, timeout, top, touch, tr, traceroute, true, tty, ttysize, udhcpc, udhcpd, udpsvd, umount, uname, uncompress, unexpand, uniq, unix2dos, unlzma, unlzop, unzip, uptime, usleep, uudecode, uuencode, vconfig, vi, vlock, volname, watch, watchdog, wc, wget, which, who, whoami, xargs, yes, zcat, zcip

Docker networking: onder de motorkap

Docker networking wordt verzorgd door de **`docker0`** bridge.

- **`docker run -it --name net1 busybox`**
  - **`ip a`** *172.17.0.2*
  - **`ip route`** *default gateway via 172.17.0.1*
  - [CTRL]+[P]+[Q]
- **`ip a s docker0`**
  - 172.17.0.1
- **`brctl show docker0`**
  - nu zien we 1 interface
- **`docker run -it --name net2 busybox`**
  - **`ip a`** *172.17.0.3*
  - **`ip route`** *default gateway via 172.17.0.1*
  - [CTRL]+[P]+[Q]
- **`brctl show docker0`**
  - nu zien we 2 interfaces

```
student@ubuntu-server-2:~$ brctl show docker0
bridge name     bridge id               STP enabled     interfaces
docker0         8000.0242297eb9c7       no              veth49f6d8b
student@ubuntu-server-2:~$ _
```

```
student@ubuntu-server-2:~$ brctl show docker0
bridge name     bridge id               STP enabled     interfaces
docker0         8000.0242297eb9c7       no              veth369e71e
                                                        veth49f6d8b
student@ubuntu-server-2:~$ _
```

Dus iedere container krijgt standaard een NIC die geconnecteerd is met de **`docker0`** virtual bridge op de host.

Docker networking: onder de motorkap

`docker attach net1`

- `ip a`
  - de netwerkkaart zit bv. in het `172.17.0.0/16` netwerk
  - we zouden dus reeds netwerk connectie moeten hebben
- `ip route`
  - `172.17.0.1` de default gateway en dit is dan ook het ip van de **`docker0`** bridge op de host
- `ping -c 5 8.8.8.8`
- `[CTRL]+[P]+[Q]`

`docker inspect net1`

- check de sectie **`NetworkSettings`**
- IP settings zijn enkel aanwezig bij running containers

```
student@ubuntu-server-2:~$ docker inspect net1 | grep -A16 NetworkSettings
        "NetworkSettings": {
            "Bridge": "",
            "SandboxID": "2cde53f5e5a423ceb5ccd5e72e2f32a275b527a47e748f130
2f8afba0fc01e13",
            "HairpinMode": false,
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "Ports": {},
            "SandboxKey": "/var/run/docker/netns/2cde53f5e5a4",
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "EndpointID": "868ffb306134919c1147c15db2eaf08099c947a2c02e604a
029a5e7709e973bf",
            "Gateway": "172.17.0.1",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
student@ubuntu-server-2:~$
```

Docker networking config files - dns

Er zijn ook container config files met netwerk settings.

`docker inspect net1 | grep resolv.conf`

`sudo cat /var/lib/docker/containers/<CONTAINERID>/resolv.conf`

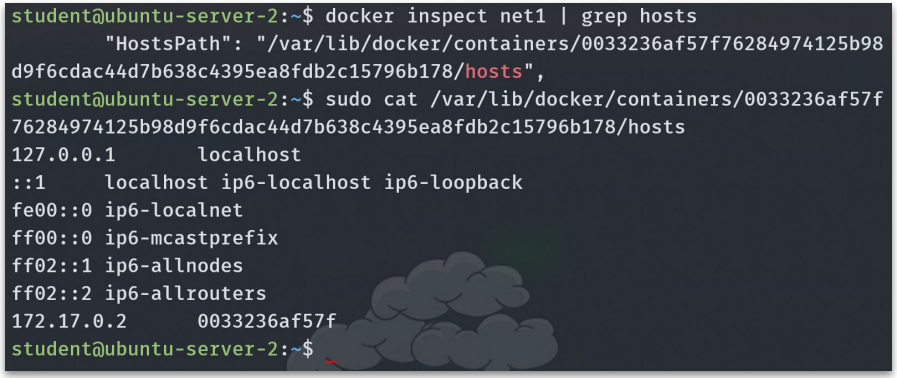
- bevat de nameserver-info
- dit is een copy van de **`resolv.conf`** op de docker-host, en wordt aangemaakt bij het opstarten van de container
- overschrijft dus de file die in de image zit
- kan aangepast worden tijdens dat de container draait, maar wordt terug gekopieerd bij het opnieuw opstarten van de container
- kan overschreven worden door een argument toe te voegen aan het `docker run` commando
- `docker run -it --dns 8.8.4.4 --name net3 busybox`
  - `cat /etc/resolv.conf`
  - `exit`
  - `docker inspect net3 | grep -wA2 Dns`
  - check de **Dns** setting

Docker networking config files - hosts

```
docker inspect net1 | grep hosts
```

```
sudo cat
/var/lib/docker/containers/<CONTAINERID>/hosts
```

- bevat de ipv4 en ipv6 static dns entries
- overschrijft dus de file die in de image zit
- kan aangepast worden in een running container, maar wordt terug gekopieerd bij het opnieuw opstarten van de container
- kan aangevuld worden door argumenten toe te voegen aan het `docker run` commando
- `docker run -it --add-host=<hostname:ip> --add-host=<hostname2:ip2> --name=net4 busybox`



Docker networking config files - hostname

`docker inspect net1 | grep hostname`

`sudo cat /var/lib/docker/containers/<CONTAINERID>/hostname`

- bevat de container id van de container
- kan overridden worden door een argument toe te voegen aan het `docker run` commando
- `docker run -it --hostname=<hostname> --name=net5 busybox`

```
student@ubuntu-server-2:~$ docker inspect net1 | grep hostname
        "HostnamePath": "/var/lib/docker/containers/0033236af57f76284974125b98d9f6cdac44d
7b638c4395ea8fdb2c15796b178/hostname",
student@ubuntu-server-2:~$ sudo cat /var/lib/docker/containers/0033236af57f76284974125b98
d9f6cdac44d7b638c4395ea8fdb2c15796b178/hostname
0033236af57f
student@ubuntu-server-2:~$ docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED             STATUS              PORTS     NAMES

0033236af57f   busybox   "sh"      38 minutes ago      Up 38 minutes               net1
54fe2aa1cefd   busybox   "sh"      About an hour ago   Up About an hour            net2
student@ubuntu-server-2:~$
```

Container port forwarding

We gebruiken een alpine-based nginx image om port forwarding te testen. Nginx is een zeer performante en lightweight webserver die speciaal geschreven is om micro-services en high-availability scenario's te ondersteunen.

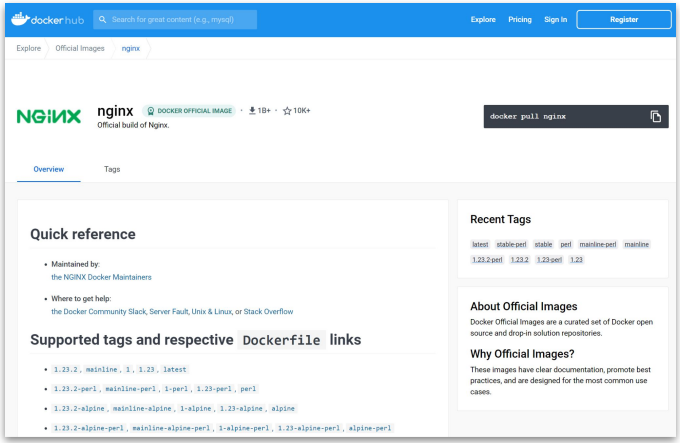
De officiële Dockerfile is gepubliceerd op github:

[docker-nginx/Dockerfile at master - alpine]

[nginx - Official Image | Docker Hub]

In de Dockerfile vinden we de `EXPOSE 80` instructie. De `EXPOSE` instructie informeert Docker dat de container luistert op de gespecificeerde netwerkpoort 80 at runtime.

Merk ook op dat nginx wordt opgestart met `-g 'daemon off;'`. Dit zorgt ervoor dat nginx op de voorgrond runt ipv als een achtergrondproces. Een container is eigenlijk een geïsoleerd proces dat op de voorgrond runt, en in deze container is dat nginx.




```
127  ENTRYPOINT ["/docker-entrypoint.sh"]
128
129  EXPOSE 80
130
131  STOPSIGNAL SIGQUIT
132
133  CMD ["nginx", "-g", "daemon off;"]
```

Container port forwarding

```
docker run -d -p 5001:80 --name web1 nginx:alpine
```

```
docker container ls
```

We zien nu in de kolom PORTS dat poort 5001 op de docker host wordt geforward naar poort 80 in de container.

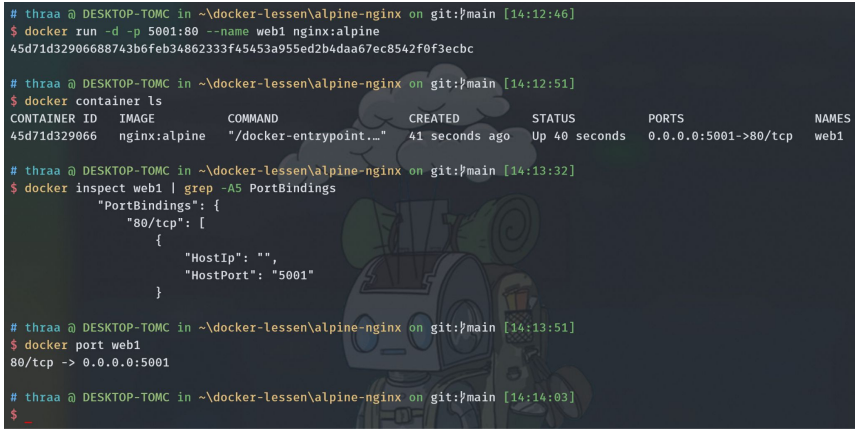
In een webbrowser surfen naar de website in de container

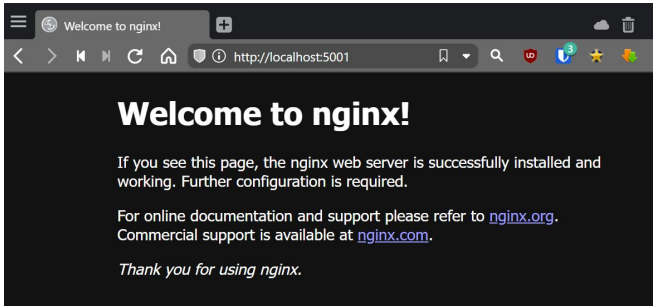
- vanaf een ubuntu docker host
  - `lynx localhost:5001`
  - surfen naar `http://<dockervm-ip>:5001`
- vanaf een Windows host
  - http://localhost:5001

```
docker inspect web1 | grep -A5 PortBindings
```

```
docker port web1
```

Hier staat de container poort wel aan de linkerkant en de docker host poort aan de rechterkant.





Container port forwarding - andere mogelijkheden

`docker run -d -p <host-ip>:5002:80 --name=web2 nginx`

- Je kan ook een poort van een ip adres (van een specifieke NIC) van de docker host forwarden naar een poort binnen een container.
- Als je nu surft naar poort 5002 van het meegegeven <dockerhost-ip> op de docker-host, kom je terecht in de container op poort 80.

`docker run -d -P --name=web3 nginx`

- Je kan alle poorten die **EXPOSED** zijn in de Dockerfile forwarden met de optie `-P`
- deze poorten worden op de host random toegewezen
- `docker port web3` toont dan welke poorten op de docker-host geforward worden. Je kan hier ook `docker container ls` voor gebruiken.
    - `80/tcp -> 0.0.0.0:12345`

`docker run -d -p 80 --name=web4 nginx`

- Je kan slechts één van de poorten die **EXPOSED** zijn in de Dockerfile forwarden met de optie `-p <PORT>`
- Deze poort wordt op de host gekoppeld aan een random poort omdat we zelf geen host poort hebben meegegeven.
- `docker port web4` toont welke poort op de docker-host geforward wordt.

Communicatie tussen containers

`docker network create --subnet 172.25.0.0/24 isolated_nw`

- We kunnen een extra netwerk toevoegen voor bepaalde containers om met elkaar te communiceren

`docker run -itd --name contain1 busybox`

- iedere container krijgt by default een IP in de docker0-bridge

`docker network connect isolated_nw contain1`

- de container krijgt nu bijkomstig een IP in het isolated_nw netwerk
- in een “user-defined”-netwerk kennen de containers elkaar via hun “hostname”

`docker run -itd --network isolated_nw --name contain2 busybox`

- deze container krijgt een IP enkel in het isolated_nw netwerk

`docker network disconnect isolated_nw contain2`

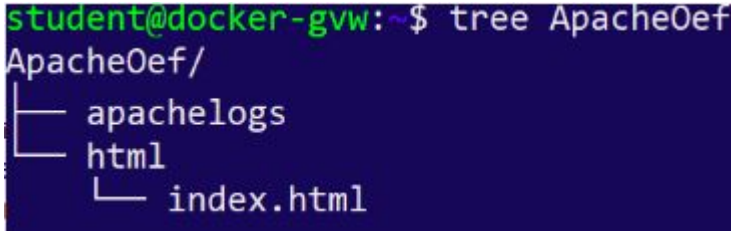
- om een container uit het netwerk te halen
- dan heeft deze container wel geen netwerk

`docker network rm isolated_nw`

- om uiteindelijk het netwerk te verwijderen

```
$ docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
dada9441de81   bridge            bridge    local
8f5ba7bc4dbd   host              host      local
327a5d3b1181   k3d-k3s-default   bridge    local
208e3e2d8286   none              null      local

# thraa @ DESKTOP-TOMC in ~ [15:32:23]
$ docker network create --subnet 172.25.0.0/24 isolated_nw
e806ff948219cc7933db31ef4f2140064c2233e38750f77cd75cbf130af7094e

# thraa @ DESKTOP-TOMC in ~ [15:32:59]
$ docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
dada9441de81   bridge            bridge    local
8f5ba7bc4dbd   host              host      local
e806ff948219   isolated_nw       bridge    local
327a5d3b1181   k3d-k3s-default   bridge    local
208e3e2d8286   none              null      local

# thraa @ DESKTOP-TOMC in ~ [15:33:05]
$ docker run -it -d --name contain1 busybox
56a5cb63f9e62f8ffdc9a5779691db040ce6ad75d5c00ab6ea8488f1dcc90a53

# thraa @ DESKTOP-TOMC in ~ [15:35:36]
$ docker network connect isolated_nw contain1

# thraa @ DESKTOP-TOMC in ~ [15:35:52]
$ docker exec -it contain1 sh
/ # ip a | grep inet
    inet 127.0.0.1/8 scope host lo
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
    inet 172.25.0.2/24 brd 172.25.0.255 scope global eth1
/ # exit

# thraa @ DESKTOP-TOMC in ~ [15:37:31]
$ docker run -it -d --network isolated_nw --name contain2 busybox
a222d12fea49c672d1d2632863f949df7905fdb6dc983b7a53d1b34adee1c42d

# thraa @ DESKTOP-TOMC in ~ [15:38:17]
$ docker exec -it contain2 sh
/ # ping -c 1 contain1
PING contain1 (172.25.0.2): 56 data bytes
64 bytes from 172.25.0.2: seq=0 ttl=64 time=0.341 ms

--- contain1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.341/0.341/0.341 ms
```

Oefening: networking & volumes



- Maak in je homefolder een directory aan, genaamd `ApacheOef`
- Zorg voor een Docker-container die
  - apacheoefening heet
  - apache runt
    - vertrekkende van de alpine linux versie van officiële image `httpd` op dockerhub
    - die poort 4444 op de host mapt naar poort 80 in de container
    - die html files krijgt van de host vanuit de volgende directory structuur
      - `ApacheOef/html/`
        - plaats hierin een index.html met de tekst: "Welkom op de SysNet-pagina van <jouw naam>"
    - die de apache logfiles plaatst op de Docker-host in de volgende folderstructuur
      - `ApacheOef/apachelogs/`

- Bekijk de website vanop de laptop door te surfen naar http://www.sysnet.lan:4444
  - hint: `hosts` file

- Bekijk de logfiles

Oefening: zelf Images maken en poorten

- Maak, via een **`Dockerfile`** zelf een image,
  - gebaseerd op de image "alpine"
  - installeer de lighttpd webserver
  - toon volgende tekst op de homepage van de website
    “Welkom op de lighttpd-homepage van <jouw naam>”
  - noem deze image mylighttpd:alpine
- Stop en verwijder alle voorgaande containers
- Start een nieuwe container gebaseerd op deze image
- Surf eens naar je webserver

De `index.html` file moet dus bij in de image komen → zie Docker_07 Dockerfile...

- Maak een nieuwe directory aan, genaamd `htdocs`. Plaats hierin een `index.html` file met een random tekst.
- Start een nieuwe container op basis van dezelfde image, maar zorg er voor dat de zojuist aangemaakte index-pagina getoond wordt.

- Zoek uit wat volgende commando's doen en test ze
  - `docker system info`
  - `docker system df`
  - `docker system prune`

Oefening: Nextcloud

- Zoek de nodige info op Docker hub om een Nextcloud container te runnen
  - gebaseerd op de image "nextcloud:latest"
  - die bereikbaar is via poort 80
  - die named-volumes gebruikt om alle data van de website en database persistent maakt
- Log aan via de website
  - **`Opgelet! De initiële configuratie via de webinterface duurt wel een tijdje`**
- Installeer ook de desktop app eens
- Voeg een file toe via de app en controleer in de web-interface en omgekeerd
- Verwijder de Container en start een nieuwe met dezelfde named-volumes. Check of alles nog werkt en aanwezig is

- Probeer hetzelfde, maar dan met host-volumes (subfolders in een nieuwe dedicated folder nextcloud)