Beskrivelse:

Denne opgave hedder Big Bang (ikke Theory). I denne opgave skal vi se på om vi kan starte alt fra opgave 2 (den med sepperat php og webcontainer) og Opgave 3 (de 3 web, med hver deres reverse proxy) på én gang. Det burde være muligt. Der er 2 muligheder. Overføre materialet til et nyt underbibliotek, eller hent data fra de andre opgaver. Begge dele er gode. For at lære lidt mere, vil jeg gennemgå det der skal til for at kunne kopiere data over fra de andre opgaver. Vi skal bruge det vi sætter op her i opgave 5.

Kapitel 1: Forberedelse.

Første skal vi, som altid, lave et nyt underbibliotek, hvori vi placerer alt vi skal bruge, og hopper ned i det (HUSK stå i dit home bibliotek (**`cd ~`**):

```
mkdir bigbang
cd bigbang
```

Eksempel output:

```
dtmek@docker2:~$ mkdir bigbang
cd bigbang
dtmek@docker2:~/bigbang$
```

Så skal vi have kopieret det vi skal bruge over til vores bibliotek. Vi starter med vores underbiblioteker fra opgave 2. Her skal vi jo have både nginx (opsætning af vores webserver), php (opsætning af php container) og vores Site (selve siden, Husk med stort S) overført. De data vi skal bruge ligger jo under cryptosep biblioteket, så det er her vi henter det, og det vil vi så gøre med 3 kommandoer (Alle tre3 kommandoer kan køre samtidig):

```
cp -r ../cryptosep/nginx/ ./
cp -r ../cryptosep/php/ ./
cp -r ../cryptosep/Site/ ./
```

Eksempel output:

```
dtmek@docker2:~/bigbang$ cp -r ../cryptosep/nginx/ ./
cp -r ../cryptosep/php/ ./
cp -r ../cryptosep/Site/ ./
dtmek@docker2:~/bigbang$
```

Så skulle det være kopieret over, lad os lige kontrollere det:

```
ls -l
```

Eksempel output:

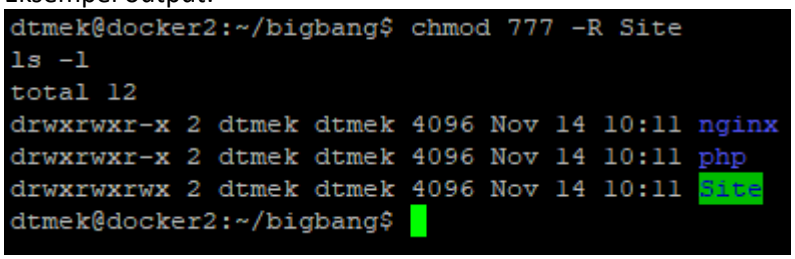
```
dtmek@docker2:~/bigbang$ ls -l
total 12
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:11 nginx
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:11 php
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:11 Site
dtmek@docker2:~/bigbang$
```

Jo de er der, men vi skal lige sætte sikkerheds på Site, ligesom vi gjorde under Opgave 1 og 2, og lige efter ser vi om det er sket.

Begge kommandoer køres samtidig:

```
chmod 777 -R Site
ls -l
```

Eksempel output:

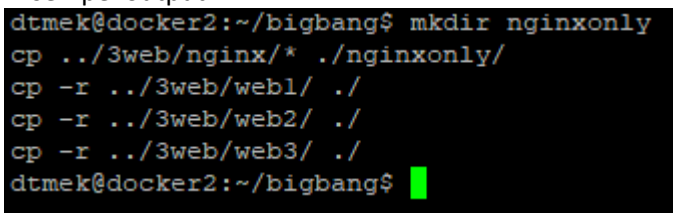


Sådan.. Så er materialerne for Opgave 2 klar. Så må vi også hellere kopiere det vi skal bruge fra Opgave 3 over. Men her løber vi jo ind i et lille problem. Under opgave 3 har vi også et bibliotek der hedder nginx. Man kan ikke have to biblioteker med samme navn ved siden af hinanden. Så det bibliotek kalder vi nginxonly i stedet for. Der skal jo ikke laves nogen ændringer ved det image. Vi må hellere komme i gang med kopiering. Der er et par biblioteker, og filer der skal kopieres.

Først kopiere vi alle de mapper der skal kopieres over. Der er 4 stk. Vi laver også lige biblioteket nginxonly (Alle kommandoer kan køres på en gang):

```
mkdir nginxonly
cp ../3web/nginx/* ./nginxonly/
cp -r ../3web/web1/ ./
cp -r ../3web/web2/ ./
cp -r ../3web/web3/ ./
```

Eksempel output:



Lad os lige se efter igen, om der er blevet kopieret noget over:

ls -l

Eksempel Output

```
dtmek@docker2:~/bigbang$ ls -l
total 28
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:11 nginx
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:47 nginxonly
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:11 php
drwxrwxrwx 2 dtmek dtmek 4096 Nov 14 10:11 Site
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:47 web1
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:47 web2
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:47 web3
dtmek@docker2:~/bigbang$
```

Alt ser ud til at være der, lad os så kopiere de filer over vi skal bruge. Det er jo konfigurationsfiler til vores reversproxy'er fra opgave 3:

```
cp ../3web/81default.conf ./
cp ../3web/82default.conf ./
cp ../3web/83default.conf ./
```

Eksempel output:

```
dtmek@docker2:~/bigbang$ cp ../3web/81default.conf ./
cp ../3web/82default.conf ./
cp ../3web/83default.conf ./
dtmek@docker2:~/bigbang$
```

Vi må hellere være sikker på at de ligger der:

```
ls -l
```

Eksempel output:

```
dtmek@docker2:~/bigbang$ ls -l
total 40
-rw-rw-r-- 1 dtmek dtmek  205 Nov 14 10:51 81default.conf
-rw-rw-r-- 1 dtmek dtmek  205 Nov 14 10:51 82default.conf
-rw-rw-r-- 1 dtmek dtmek  205 Nov 14 10:51 83default.conf
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:11 nginx
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:47 nginxonly
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:11 php
drwxrwxrwx 2 dtmek dtmek 4096 Nov 14 10:11 Site
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:47 web1
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:47 web2
drwxrwxr-x 2 dtmek dtmek 4096 Nov 14 10:47 web3
dtmek@docker2:~/bigbang$
```

Jo de ligger der. Så skal vi kun have lavet vores docker-compose.yml (HUSK: Alt med små bogstaver!). Det er så næste kapitel.

Slut på Kapitel 1: Forberedelse.

Kapitel 2: docker-compose.yml

Jeg vil ikke gennemgå de forskelille dele af filen denne gang. Det er kun sammenlægning af de filer der findes i opgave 2, og 3. Se disse eksempler for gennemgang. Da vi gerne vil gemme vores opsætning efter dette. Laver vi med det samme imagenavnene så vi kan pushe senere. Først åbner vi vores editor (HUSK: ALT med småt):

```
nano docker-compose.yml
```

Filen skal indeholde følgende (Kopier teksten herunder, og sæt det ind i dit SSH vindue ved at højreklikke i vinduet) HUSK at indsætte IP på din registry server på de fremhævede steder **VIGTIGT**!!! At alle indrykninger skal være som herunder!!!! Der anbefales at kopier filen fra det materiale der kan downloades:

```
######### docker-compose.yml start ##########################
services:
# Lav maskine fra Lektion 2
   # Lav Web server (nginx)
  nginx:
    image: ip_Registry_server:5000/web:latest
    build: ./nginx/
    hostname: web
    container_name: web
    restart: unless-stopped
    ports:
      - 80:80
    volumes:
        - ./Site/:/var/www/html/
    depends_on:
      - php

   # Lav en PHP service
  php:
    image: ip_Registry_server:5000/php:latest
    build: ./php/
    hostname: php
    container_name: php
    restart: unless-stopped
    expose:
      - 9000
    volumes:
       - ./Site/:/var/www/html/

# Lav maskiner fra lektion 3
  web1:
    image: ip_Registry_server:5000/web1:latest
    build: ./nginxonly/
    hostname: web1
    container_name: web1
    restart: unless-stopped
    expose:
      - 80
    volumes:
```

```
    - ./web1:/usr/share/nginx/html
  networks:
    web:
      ipv4_address: 192.168.180.81

web2:
  image: ip_Registry_server:5000/web2:latest
  build: ./nginxonly/
  hostname: web2
  container_name: web2
  restart: unless-stopped
  expose:
    - 80
  volumes:
    - ./web2:/usr/share/nginx/html
  networks:
    web:
      ipv4_address: 192.168.180.82

web3:
  image: ip_Registry_server:5000/web3:latest
  build: ./nginxonly/
  hostname: web3
  container_name: web3
  restart: unless-stopped
  expose:
    - 80
  volumes:
    - ./web3:/usr/share/nginx/html
  networks:
    web:
      ipv4_address: 192.168.180.83

proxy1:
  image: ip_Registry_server:5000/proxy1:latest
  build: ./nginxonly/
  hostname: proxy1
  container_name: proxy1
  restart: unless-stopped
  ports:
    - 81:80
  volumes:
    - ./81default.conf:/etc/nginx/conf.d/default.conf
  networks:
    web:
      ipv4_address: 192.168.180.71

proxy2:
  image: ip_Registry_server:5000/proxy2:latest
  build: ./nginxonly/
  hostname: proxy2
  container_name: proxy2
  restart: unless-stopped
```

```
    ports:
      - 82:80
    volumes:
      - ./82default.conf:/etc/nginx/conf.d/default.conf
    networks:
      web:
        ipv4_address: 192.168.180.72

  proxy3:
    image: ip_Registry_server:5000/proxy3:latest
    build: ./nginxonly/
    hostname: proxy3
    container_name: proxy3
    restart: unless-stopped
    ports:
      - 83:80
    volumes:
      - ./83default.conf:/etc/nginx/conf.d/default.conf
    networks:
      web:
        ipv4_address: 192.168.180.73

networks:
  web:
    name: web
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.180.0/24
################## docker-compose.yml slut #############
```

Gem filen ved at trykke ”Ctrl+x” -> trykke ”Y” -> Tryk Enter ved filnavn. Og du er tilbage til normal promt på Ubuntu.

Det var jo egentlig bare at sætte de to filer sammen. Men lad os da se om det virker. Vi starter lige vores containere, der er defineret:

`docker compose up -d`

Eksempel output:

```
dtmek@docker2:~/bigbang$ docker compose up -d
[+] Running 28/28
 ✔ php Pulled
   ✔ a480a496ba95 Pull complete
   ✔ a47alde29151 Pull complete
   ✔ a0821bbad4e4 Pull complete
   ✔ 6be174f186fb Pull complete
   ✔ bd4787ab9f9a Pull complete
   ✔ 741ae76dd1e2 Pull complete
   ✔ 7e6f32953ef1 Pull complete
   ✔ 15b2eladddd6 Pull complete
   ✔ 6c8a58b73ea3 Pull complete
   ✔ 4f4fb700ef54 Pull complete
   ✔ 79e003456a94 Pull complete
   ✔ 6ba217340f80 Pull complete
   ✔ 2c3e38f70930 Pull complete
 ✔ proxy1 Pulled
 ✔ web2 Pulled
   ✔ f3ace1b8ce45 Pull complete
   ✔ 11d6fdd0e8a7 Pull complete
   ✔ f1091da6fd5c Pull complete
   ✔ 40eea07b53d8 Pull complete
   ✔ 6476794e50f4 Pull complete
   ✔ 70850b3ec6b2 Pull complete
 ✔ web3 Pulled
 ✔ web1 Pulled
 ✔ proxy2 Pulled
 ✔ nginx Pulled
   ✔ 4165791b2583 Pull complete
 ✔ proxy3 Pulled
[+] Running 10/10
 ✔ Network web              Created
 ✔ Network bigbang_default  Created
 ✔ Container proxy3         Started
 ✔ Container web1           Started
 ✔ Container web3           Started
 ✔ Container proxy1         Started
 ✔ Container web2           Started
 ✔ Container proxy2         Started
 ✔ Container php            Started
 ✔ Container web            Started
dtmek@docker2:~/bigbang$
```

Der kan man bare se! Den gjorde ikke andet end af pulle det hele fra vores Registry! Det var jo smart. Den behøvede ikke at bygge det hele. Den hentede det bare. Så nu kan man også se at det måske er en god ide at have et registry! Men der er lige en lille ting.... Prøv at se på resultatet til sidst, der er et netværk der hedder bigbang_default? Lige ved siden af vores web netværk?

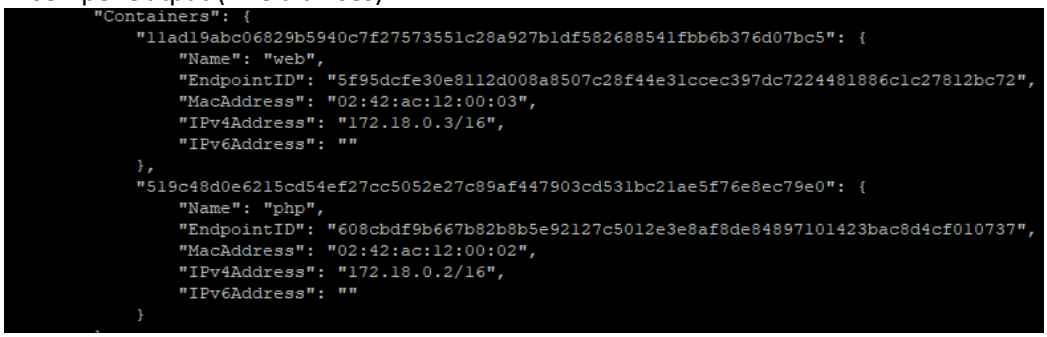
```
 ✔ Network web              Created
 ✔ Network bigbang_default  Created
 ✔ Container proxy3         Started
 ✔ Container web1           Started
 ✔ Container web3           Started
 ✔ Container proxy1         Started
 ✔ Container web2           Started
 ✔ Container proxy2         Started
 ✔ Container php            Started
 ✔ Container web            Started
dtmek@docker2:~/bigbang$
```

Hvad er dog det for en fisk?

Lad os lige kigge hvilke containere der sidder i den:

`docker network inspect bigbang_default`

Eksempel Output (Ikke alt vises):



AHA (Og nej ikke pop gruppen...) det er de to containere fra opgave 2. Dem havde vi jo heller ikke givet en ip adresse. Så er det opklaret. Docker har bare oprettet et dummyweb til dem. Lad os lige se om alle vores containere kører. Og de vil også køre efter en genstart, da alle er sat til **`restart: unless-stopped`** : i vores docker-compose.yml fil:

`docker ps`

Eksempel output:

```
dtmek@docker2:~/bigbang$ docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS          PORTS                                      NAMES
11ad119abc068   192.168.1.131:5000/web:latest      "/docker-entrypoint.…"   31 minutes ago   Up 31 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp          web
519c48d0e621   192.168.1.131:5000/php:latest      "docker-php-entrypoi…"   31 minutes ago   Up 31 minutes   9000/tcp                                   php
5c06e69701e0   192.168.1.131:5000/web1:latest     "/docker-entrypoint.…"   31 minutes ago   Up 31 minutes   80/tcp                                     web1
a61d50d91700   192.168.1.131:5000/web2:latest     "/docker-entrypoint.…"   31 minutes ago   Up 31 minutes   80/tcp                                     web2
db66bbd0c87b   192.168.1.131:5000/proxy3:latest   "/docker-entrypoint.…"   31 minutes ago   Up 31 minutes   0.0.0.0:83->80/tcp, [::]:83->80/tcp        proxy3
71d248716bf9   192.168.1.131:5000/proxy2:latest   "/docker-entrypoint.…"   31 minutes ago   Up 31 minutes   0.0.0.0:82->80/tcp, [::]:82->80/tcp        proxy2
82d35523069c   192.168.1.131:5000/proxy1:latest   "/docker-entrypoint.…"   31 minutes ago   Up 31 minutes   0.0.0.0:81->80/tcp, [::]:81->80/tcp        proxy1
0fac1306d897   192.168.1.131:5000/web3:latest     "/docker-entrypoint.…"   31 minutes ago   Up 31 minutes   80/tcp                                     web3
dtmek@docker2:~/bigbang$
```

De kører jo alle sammen. Hvilket egentlig er meget godt. De skal nemlig bruges i næste opgave der handler om den grafiske brugerflade portainer.io. Så denne gang er der intet der skal fjernes eller lukkes.

Slut på Kapitel 2: docker-compose.yml

Og også slut på Opgave 4.