

Beskrivelse:

Dette er en gennemgang af den grafiske brugerflade der hedder portainer.io, eller bare kaldet for portainer. Modsat alle andre eksempler vil vi denne gang IKKE lave et nyt separat underbibliotek at arbejde i, da vi også skal bruge data fra opgave 4. Så vi arbejder kun i vores opgave 4 directory. Lad os komme i gang:

Kapitel 1: Forberedelse.

Først skal vi, hvis du ikke står i opgave 4's bibliotek, ned i det:

```
cd bigbang
```

Eksempel Output:

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

Vi skal også lige have lavet lidt forberedelse inden vi kaster os ud i Portainer. Vi skal have lavet et underbibliotek, med en fil i, og en ekstra fil. Først laver vi underbiblioteket, og hoppe ned i det (begge kommandoer kan køres samtidig):

```
mkdir web4  
cd web4
```

Eksempel output:

```
dtmek@docker2:~/bigbang$ mkdir web4  
cd web4  
dtmek@docker2:~/bigbang/web4$ █
```

nu åben en ny fil med nano:

```
nano index.html
```

Indhold til filen på næste side.

Filen skal indeholde følgende (Kun det markerede, UDEN start og slut mærket!):

```
##### index.html #####
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="da"><head><title>TOP SECRET 4!!!</title>
```

```
<meta charset="utf-8">
```

```
<meta content="text/html; charset=UTF-8" http-equiv="content-
type"></head><body>
<big><big style="font-family: monospace; color:
red;"><big><big><big><big>TOP
SECRET PAGE!!!<br>
Nr 4 (FIRE!!)<br>
Læs ikke denne tekst<br>
Denne tekst er hemelig!</big></big></big></big></big></big>
<br><big><big><br>
</big></big><br>
</body></html>
```

```
##### index.html slut #####
```

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

Derefter hop tilbage til forrige bibliotek:

```
cd ..
```

Eksempel output:

```
dtmek@docker2:~/bigbang/web4$ cd ..
dtmek@docker2:~/bigbang$
```

Her skal vi også lige have lavet en fil:

```
nano 84default.conf
```

Indhold af filen på næste side.

Filen skal indeholde følgende:

```
##### 84default.conf #####  
# Complete Nginx Docker reverse proxy config file  
server {  
    listen 80;  
  
    location / {  
        proxy_pass http://192.168.180.84:80/index.html;  
    }  
  
} # End of Docker Nginx reverse proxy example file  
##### 84default.conf  slut #####
```

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

Det vi har lavet nu skal bruges senere, når vi prøver at oprette containere med portainer.

Slut på Kapitel 1: Forberedelse.

Kapitel 2. install Portainer.

Nu skal vi have installeret portainer, Det gode ved det, er at portainer faktisk også er en Docker container! Så vi skal bare have startet den container, så har vi portainer kørende... Men først skal vi have oprettet et dataområde til portainer.. Et dataområde er en slags harddisk, som udelukkende bliver styret af docker. Dvs det er ikke afhængig af at have et bibliotek struktur på din host. Men lad os oprette det:

```
docker volume create portainer_data
```

Forklaring:

docker volume: Dette er kommandoen for volumes.

Create: Opret en volume.

Portainer_data: Navnet på vores volume. I dette tilfælde **Portainer_data**

Eksempel output:

```
dtmek@docker2:~/bigbang$ docker volume create portainer_data
portainer_data
dtmek@docker2:~/bigbang$
```

Så har vi lavet et data volume, som Portainer har brug for. Vi må hellere starte den portainer container:

```
docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:2.21.4
```

Forklaring:

Det var en lang linje! Der er små forskelle fra de andre containere vi har startet med docker run. Her har vi flere porte åbne, og vi mounter portainer_data til sidst, sammen med at vi starter og henter containeren (containeren vi henter hedder: **portainer-ce:2.21.4** Og det indeholder navn:version.).

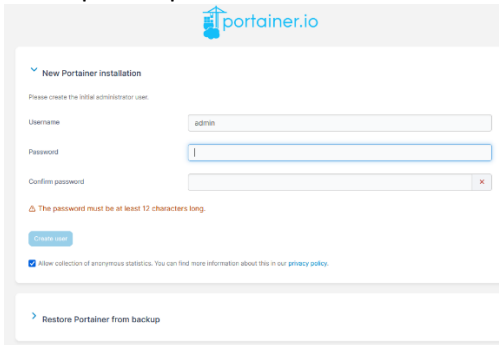
Eksempel output:

```
dtmek@docker2:~/bigbang$ docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:2.21.4
Unable to find image 'portainer/portainer-ce:2.21.4' locally
2.21.4: Pulling from portainer/portainer-ce
2a8c27161aa3: Pull complete
679061c20821: Pull complete
d40df14c1d7a: Pull complete
8215717c7c10: Pull complete
342665f0e7c: Pull complete
8c27c7448b34: Pull complete
070d3bf2828e: Pull complete
846480e9f8b0: Pull complete
c7053d7d4c2a: Pull complete
a2ed6de7f05f: Pull complete
444b700c2f54: Pull complete
Digest: sha256:42a7f5abd4735f9cd91563c6134e014b15168c4018b6ea87f1eac9d9618b2add
Status: Downloaded newer image for portainer/portainer-ce:2.21.4
4073b0ebf1b1d322f0c08052cdd282d53087eb3430c09a29c74fb0e3ce74f701
dtmek@docker2:~/bigbang$
```

Nu skal vi i gang med GUI. Vi går til portainer containerens webside via webbrowser: Der vil muligvis komme en sikkerhedsadvarsel da vi skriver https: Du går bare videre:

https://ip_på_worker_makine:9443

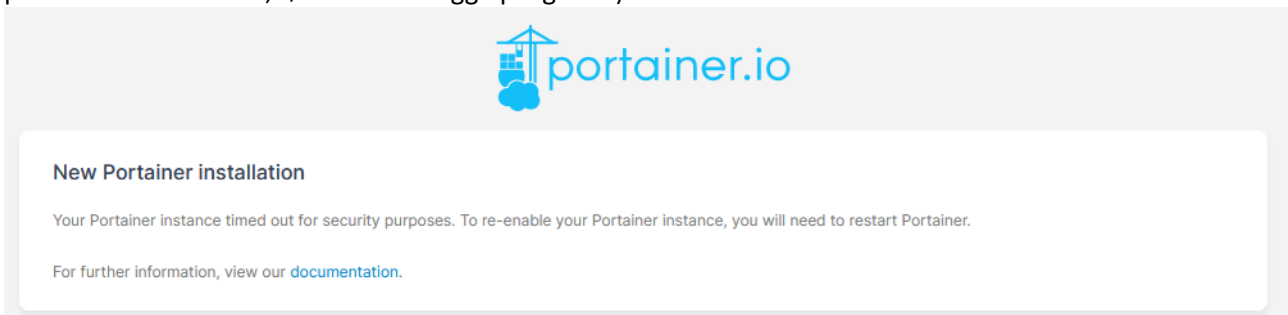
Eksempel Output:



Vi skal først oprette en admin bruger, og indtaste et password. Bruger lader vi være admin, og password er 123456789012. Man må selv finde ud af om man vil have kryds i følgende, jeg fjerner krydset:

Allow collection of anonymous statistics. You can find more information about this in our [privacy policy](#).

Og tryk på "create user" knappen. Hvis man er for langsom, kommer følgende frem, her skal man genstarte portainer containeren, før man kan logge på igen.. :)



Genstart container gøres ved at skrive:

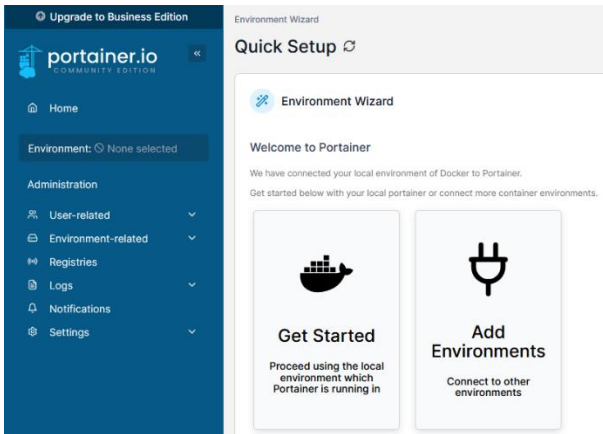
docker container restart portainer

Eksempel output:

```
dtmek@docker2:~/bigbang$ docker container restart portainer
portainer
dtmek@docker2:~/bigbang$
```

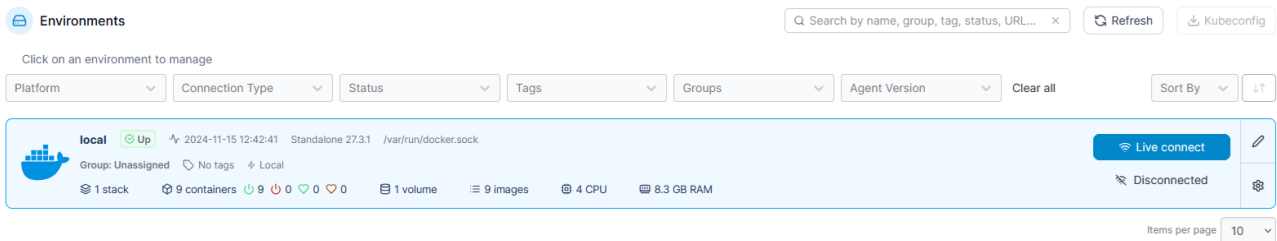
Så kan du prøve vi igen og gå til **https://ip_på_worker_makine:9443**, og indtaste info lidt hurtigere end der blev gjort sidst....

Så er vi i gang:

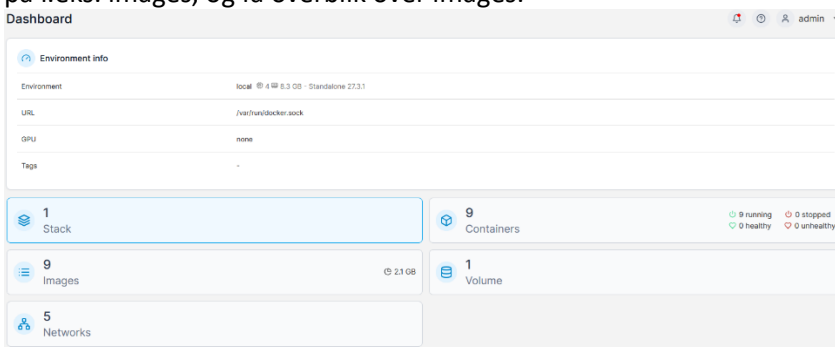


Først skal vi lige have forbindelse til vores lokale docker system:

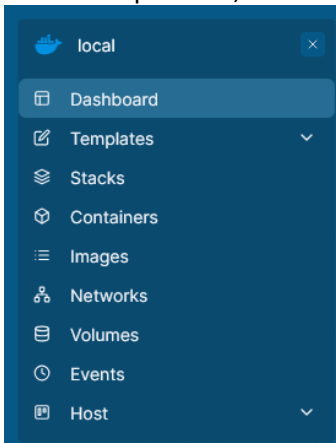
Tryk på **Home** lude til venstre:



Her kan vi se vores lokale Docker system, og hvilke ressourcer der er, og hvor mange containere der kører mm. Tryk på den knap til højre der hedder **Live connect** Her ser vi vores Dashboard. Man kan klikke på f.eks. images, og få overblik over images:



Til venstre på siden, er der også mulighed for at kigge nærmere på ting:



Hvis i vil, kan i prøve at klikke lidt rundt, inden vi går videre. Hvis i ikke ved hvordan i kommer videre, eller bare er låst fast, luk webbrowser, og connect til websiden igen. Jeg vil også lige nævne at Templates viser alle de standard images vi kan hente. Det skal vi ikke bruge denne gang. Det kan i lege med efter opgave 6 :) Alle andre punkter viser informationer vi indtil nu kun har set på liste form. F.eks. tryk på "Containers" viser det samme som kommandoen **docker container ls -a**.

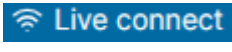
Slut på Kapitel 2. install Portainer.


Kapitel 3. Portainer GUI

Vi kan selvfølgelig også oprette containere. Så nu vil vi oprette en webseite (web4) og den dertilhørende reverse proxy server. Og det gør vi kun med Portainer denne gang.

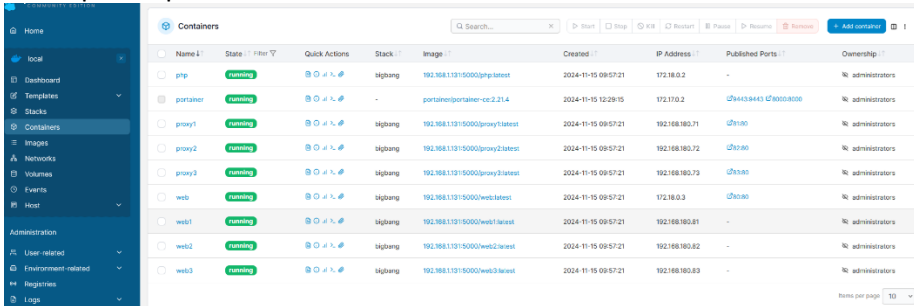
Åben websiden for portainer, hvis den ikke allerede er åben:

https://ip_på_worker_makine:9443


og log på (brugernavn: admin og password: 123456789012), og tryk  til din lokale server.

For at oprette containere i Portainer, behøve vi ikke oprette et image først. Så vi går direkte i gang, ved på venstre side, at vælge .

Eksempel output:



Name	State	Quick Actions	Stack	Image	Created	IP Address	Published Ports	Ownership
php	Running		sjbang	192.168.1.121:5000/php:latest	2024-11-15 09:57:21	172.18.0.2	-	No administrators
portainer	Running		-	portainer/portainer-ce:2.21.4	2024-11-15 12:29:15	172.17.0.2	0.0.0.0:8000->8000	No administrators
proxy1	Running		sjbang	192.168.1.121:5000/proxy1:latest	2024-11-15 09:57:21	192.168.180.71	0.0.0.0	No administrators
proxy2	Running		sjbang	192.168.1.121:5000/proxy2:latest	2024-11-15 09:57:21	192.168.180.72	0.0.0.0	No administrators
proxy3	Running		sjbang	192.168.1.121:5000/proxy3:latest	2024-11-15 09:57:21	192.168.180.73	0.0.0.0	No administrators
web	Running		sjbang	192.168.1.121:5000/web:latest	2024-11-15 09:57:21	172.18.0.3	0.0.0.0	No administrators
web1	Running		sjbang	192.168.1.121:5000/web1:latest	2024-11-15 09:57:21	192.168.180.81	-	No administrators
web2	Running		sjbang	192.168.1.121:5000/web2:latest	2024-11-15 09:57:21	192.168.180.82	-	No administrators
web3	Running		sjbang	192.168.1.121:5000/web3:latest	2024-11-15 09:57:21	192.168.180.83	-	No administrators

Nu trykkes der på knappen  øverst til højre i billedet, og man kommer til en side der hedder **Create container**.

Følgende pladser skal udfyldes med følgende informationer:

Name: Dette er navnet på containeren, og her vil det være "web4":


Eksempel output:

Name

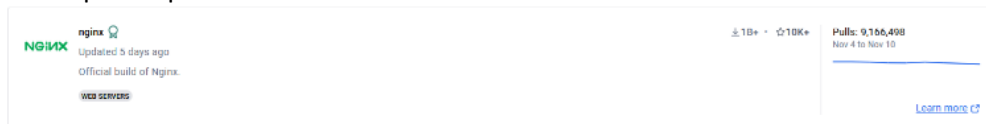
Derefter skal "Image configuration" sættes op, hvad er vores parrent image? Jo i dette tilfælde vil det være nginx(:latest) da det jo skal være en webserver:

Eksempel output:


Image Configuration
Registry
Image

Hvis man ikke er helt sikker på om det image man har skrevet eksisterer, kan man lave et opslag ved at trykke på  yderst til højre, her vil den komme med forslag på images, den der passer bedst kommer øverst. Vi kan da lige prøve, så tryk på "search", og se hvad der kommer øverst:

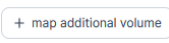
Eksempel output:



Som man kan se er den officielle nginx øverst. Så bare luk søgefanebladet igen, og vi er tilbage til Portainer. Resultatet bliver ikke overført! Det næste vi skal rette til i vores container, det er nede i området der hedder

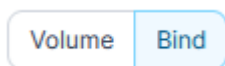
 Advanced container settings :

Du vælger fanebladet  :

Og tryk på 

Til højre for feltet "container" linjen vælges "Bind"

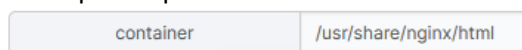
Eksempel output:



I feltet "container" skrives:

/usr/share/nginx/html

Eksempel output:



I Host skrives KOMPLET Sti, for at finde denne sti vender vi tilbage til vores SSH forbindelse til vores Docker work maskine. Vi skal selvfølgelig stå i vores bigbang bibliotek, for ellers ville vi jo ikke have stien til hvor web4 ligger. Kommandoen til dette er **pwd** (det betyder present work directory):

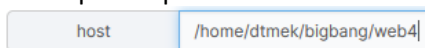
pwd

Eksempel output:

```
dtmek@docker2:~/bigbang$ pwd
/home/dtmek/bigbang
dtmek@docker2:~/bigbang$
```

Denne sti taster vi ind i feltet "host", husk at gemme denne sti, den skal også bruges til vores proxy container, og husk at tilføje **/web4** til sidst i stien. Det er jo her vores webside ligger.

Eksempel output:



Du vælger nu fanebladet:

Network

I øverste felt vælges netværket "web" via dropdown menuen.

Eksempel output:

Network

web

I "Hostname" skrives vores maskines navn (web4):

Eksempel output:

Hostname

web4

I "IPv4 address" skrives vores ip adresse til web4 (192.168.180.84):

Eksempel output:

IPv4 Address

192.168.180.84

Nu vælges fanebladet:

Restart policy

Her vælges "Unless stopped"

Eksempel output:

Never

Always

On failure

Unless stopped

Nu er vi klar til at oprette containeren. Det gør vi ved at trykke på knappen

Deploy the container

Der er lige

over



Advanced container settings



Funktion:

Knappen ændrer sig kortvarigt til at vise

Deployment in progress...

Og derefter skifter visningen til Container list.

Og her kan vi finde vores container:

Eksempel output:

web4 running nginx:latest 2024-11-18 08:55:14 192.168.180.84 administrators

Nu kører vores container, men vi kan jo ikke se den uden også at have en reverse proxy. De fleste ting skal udfyldes som ovenstående. Men først går vi til [+ Add container](#) :

Name er navnet på container (proxy4)

Eksempel output:

Name

Det bygger igen over et nginx image:

Eksempel output:

Image Configuration
Registry
Image

Denne gang skal vi også have **Network ports configuration** sat op. Vi skal jo have porten 84 på hosten overført til port 80 på containeren, så tryk på [+ map additional port](#) (Husk at vælge tcp trafik, hvis man også skulle have udp, skulle der oprettes endnu en linje, ved at trykke endnu engang på [+ map additional port](#)):

Eksempel output:

Network ports configuration
Publish all exposed ports to random host ports
Port mapping →
[+ map additional port](#)

Du vælger fanebladet [Volumes](#) under [Advanced container settings](#) :

Og tryk på [+ map additional volume](#)

Til højre for feltet "container" linjen vælges "Bind"

Eksempel output:

I feltet "container" skrives:

`/etc/nginx/conf.d/default.conf`

Eksempel output:

Nu da det er en config fil, skal den jo mountes Read only. Dette gør vi ved for enden af feltet "host" at ændre filen til Read-only:

Eksempel output:

Writable Read-only

Vi fandt jo vores pwd sidst vi skulle bruge den, så den bruges igen, men vi skal også huske at indsætte vores fil i vores sti. Det er jo kun en fil vi mounter nu.

Eksempel output:

→

Du vælger nu fanebladet: Network

Her indtastes følgende:

Network : web
Hostname: proxy4
IPv4 Address: 192.168.180.74

Eksempel output:

Network	<input type="text" value="web"/>
Hostname	<input type="text" value="proxy4"/>
Domain Name	<input type="text" value="e.g. example.com"/>
MAC Address	<input type="text" value="e.g. 12-34-56-78-9a-bc"/>
IPv4 Address	<input type="text" value="192.168.180.74"/>

Nu vælges fanebladet: Restart policy

Her vælges "Unless stopped"

Eksempel output:

Never Always On failure Unless stopped

Igen er vi nu klar til at oprette containeren. Det gør vi ved, igen, at trykke på knappen , der er lige over Advanced container settings :

Funktion:

Knappen ændrer sig kortvarigt til at vise Og derefter skifter visningen til Container list. Og her kan vi finde vores container (Der kan være flere sider, nederst på siden, kan man vælge det sidenummer man vil se):

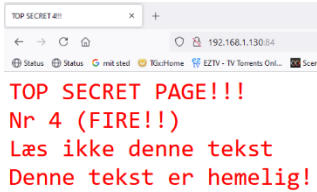
Eksempel output:

<input type="checkbox"/> proxy4	<input checked="" type="checkbox"/> running	<input type="checkbox"/> .il > .	nginx:latest	2024-11-18 09:26:40	192.168.180.74	84:80	administrators
---------------------------------	---	----------------------------------	--------------	---------------------	----------------	-------	----------------

Nu har vi både webserver 4, og proxy4 kørende, så lad os besøge den webside vi lige har oprettet:

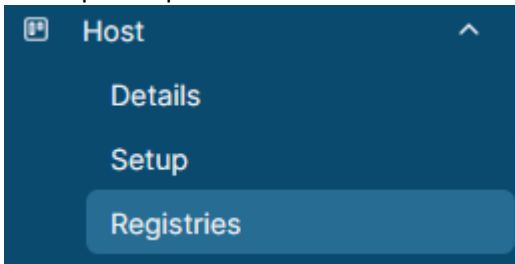
http://ip_på_din_maskine:84

Eksempel output:

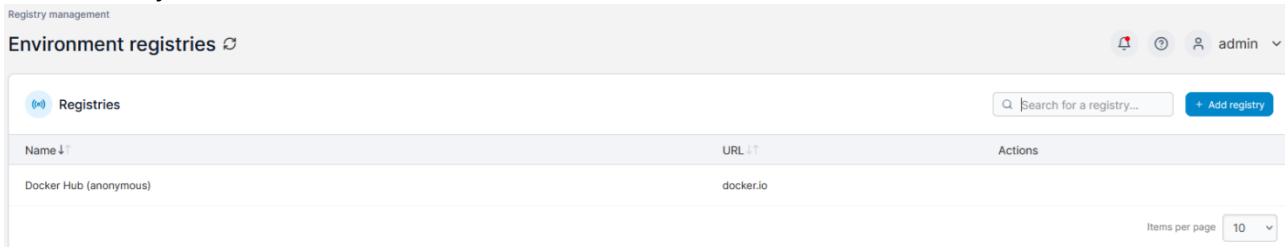


Det virkede! Vi kan ikke gemme vores containere, eller mere vores image til vores registry, uden at begynde på de mere avancerede funktioner, men det kan vi jo allerede gøre uden brug af GUI (Push!). Desuden har vi jo ikke kaldt vores image et navn med IP adressen på vores Registry. Vi kan så i stedet for pulle et image vi har lavet fra vores registry. Her er der ikke så meget der skal laves, så lad os pulle det allerførste image vi har lavet. Det er det image der indeholder php og web i én container. Først skal vi have oprettet vores Registry server. Det gør vi ved ude til venstre at gå til "Host -> Registries":

Eksempel output:



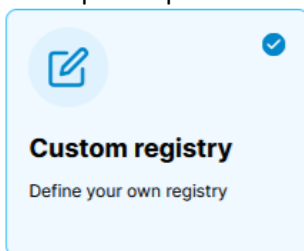
Sådan ser højre side ud af billedet:



Tryk så på **+ Add registry** knappen oppe til højre:

Vælg yderst til højre "Custom registry"

Eksempel output:



Derefter udfyldes resten med informationerne:

Navn: Minregistry

Registry URL: ip_på_din_Registry:5000

Eksempel Output:

Custom registry details

Name*

Registry URL ⓘ*

Authentication ⓘ

Derefter trykkes der på **Add registry** og registry er tilføjet

Eksempel output:

Name ↓↑	URL ↓↑	Actions
<input type="checkbox"/> Docker Hub (anonymous)	docker.io	<input type="button" value="Hide for all users"/> <input type="button" value="Business Feature"/>
<input type="checkbox"/> Minregistry	192.168.1.131:5000	<input type="button" value="Browse"/> <input type="button" value="Business Feature"/>

Vær opmærksom på at man IKKE kan bruge , uden at betale for Portainer!

Vi har nu opsat vores registry, og vi kan hente fra den. Vi starter med igen at gå til Containeres ude til venstre, og trykke **+ Add container** :

”Name” laver vi til crypto:

Eksempel output:

Name

Under Registry vælger vi med dropdown menuen Minregistry, og skriver ”crypto” under image:

Eksempel output:

Image Configuration

Registry

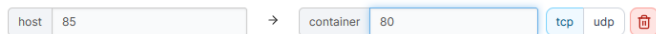
Image

Vi går ned til afsnittet: **Network ports configuration** og trykker på: + map additional port

Vi åbner nu porten 85 til containeres port 80:

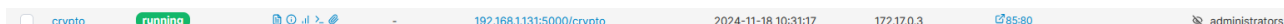
Eksempel output:

Port mapping ⓘ



Så skal der ikke laves mere, alt ligger jo i containeren. Så tryk på Deploy the container og vores crypto bliver hentet og kører:

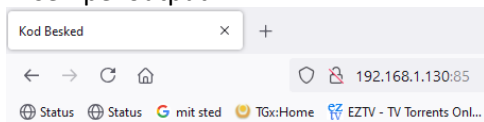
Eksempel output:



Prøv at gå til adressen:

http://ip_på_din_maskine:85

Eksempel output:



TOP SECRET PAGE!!!

Tryk på en af følgende opgaver:

- [1. Kod din besked](#)
- [2. DEkod din besked](#)

Så er vi færdige med at gennemgå Portainer.io. Man er fri til at lege lidt med Portainer, inden vi går til den Sidste demonstration.

Efter du er færdige med at lege, skal vi have slette alle container, images og Netværket. Bare rolig. Vi kopierer alt over i vores maskine i næste opgave. Der skal ikke tages ret meget. Men først skal vi lige have ryddet op:

Hvis du er på virtuelle maskiner, restore vores arbejdsmaskine (Worker) til lige efter vi har installeret docker. Til det punkt hvor jeg skrev: **Dette er et godt tidspunkt at lave et snapshot, hvis du er på en virtuel maskine. Det gør det nemmere i allersidste opgave!**

Ellers må du i gang med:

```
docker container rm container1 container2 container3 osv.  
docker image rm image1 image2 image 3 osv.  
Docker network rm net1 osv.  
osv.
```

Slut på Kapitel 3. Portainer GUI

Og slut på opgave 5.