

## Ukesoppgaver 10 – Hyperthreading og Docker

**RØD** - Obligoppgaver

**GUL** – Ikke obligoppgaver

**TURKIS** – Ukens nøtt og utfordringer

### 1. (OBLIG)

Under er vist at Linux-VM har thread\_siblings:

```
/sys/devices/system/cpu/cpu9/topology/thread_siblings_list:9,57
/sys/devices/system/cpu/cpu90/topology/thread_siblings_list:42,90
/sys/devices/system/cpu/cpu91/topology/thread_siblings_list:43,91
/sys/devices/system/cpu/cpu92/topology/thread_siblings_list:44,92
/sys/devices/system/cpu/cpu93/topology/thread_siblings_list:45,93
/sys/devices/system/cpu/cpu94/topology/thread_siblings_list:46,94
/sys/devices/system/cpu/cpu95/topology/thread_siblings_list:47,95
[group25@os25:~]$ grep "" /sys/devices/system/cpu/cpu*/topology/thread_siblings_list | wc -l
96
[group25@os25:~]$
```

### 2. (OBLIG)

Under legger jeg til skriptet og endrer timeformat for å få det ryddigere. Også kompilerer jeg med gcc og kjører med time:

```
[group25@os25:~]$ cat mem.c
/*oppgave 2 uke 10 */

#include <stdio.h>

int array[102400];

void main(){
    int i,k;
    for(k=0;k<2000000;k++)
    {
        for(i = 0;i < 1024;i++){
            array[i] = i;
        }
    }
}

[group25@os25:~]$ TIMEFORMAT="Real:%R User:%U System:%S %P%"
[group25@os25:~]$ gcc mem.c
[group25@os25:~]$ time ./a.out
Real:10.004 User:10.003 System:0.000 99.99%
```

Under runner jeg to ganger parallellt på samme CPU:

```
group25@os25:~$ for i in {1..2}; do time taskset -c 0 ./a.out & done
[1] 108251
[2] 108252
group25@os25:~$
```

regn skal kjøres på CPU 0.

```
amnadastgir — group25@os25: ~ — ssh group25@os25.vlab.cs.oslomet.no...

top - 17:24:45 up 24 days, 23:42, 2 users, load average: 2.86, 2.77, 2.81
Tasks: 72 total, 3 running, 69 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.3 us, 0.1 sy, 0.0 ni, 97.6 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 773178.5 total, 648177.8 free, 28349.8 used, 96651.0 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 739964.6 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND	P
108254	group25	20	0	2756	580	512	R	33.6	0.0	0:02.56	a.out	0
108253	group25	20	0	2756	576	512	R	33.2	0.0	0:02.55	a.out	0
1	root	20	0	2608	596	528	S	0.0	0.0	0:00.05	sh	65

Hvis jeg bestemmer at begge skal kjøres på CPU 0:

```
group25@os25:~$ Real:31.024 User:10.358 System:0.000 33.38%
Real:31.025 User:10.357 System:0.000 33.38%
^C
[1]- Exit 255          time taskset -c 0 ./a.out
[2]+ Exit 255          time taskset -c 0 ./a.out
```

Hvis operativsystemet velger selv da bruker den to ulike CPU-er: og det tar omtrent 66,6% raskere enn da jeg bestemte CPU.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND	P
108775	group25	20	0	2756	576	512	R	100.0	0.0	0:07.07	a.out	4
108776	group25	20	0	2756	516	448	R	100.0	0.0	0:07.08	a.out	8

```
group25@os25:~$ Real:10.014 User:10.001 System:0.000 99.86%
Real:10.019 User:9.998 System:0.004 99.83%
^C
[3]- Exit 255          time ./a.out
[4]+ Exit 255          time ./a.out
```

Det som skjer her, er hyperthreading:

Hyperthreading (også kjent som simultaneous multithreading) er en teknologi som lar en enkelt fysisk prosessor kjerne behandle flere tråder samtidig ved å dele opp den fysiske prosessoren i virtuelle prosessorer, kalt «logiske prosessorer». Dette kan gi en betydelig økning i systemets ytelse, spesielt for applikasjoner som bruker mye av prosessoren som

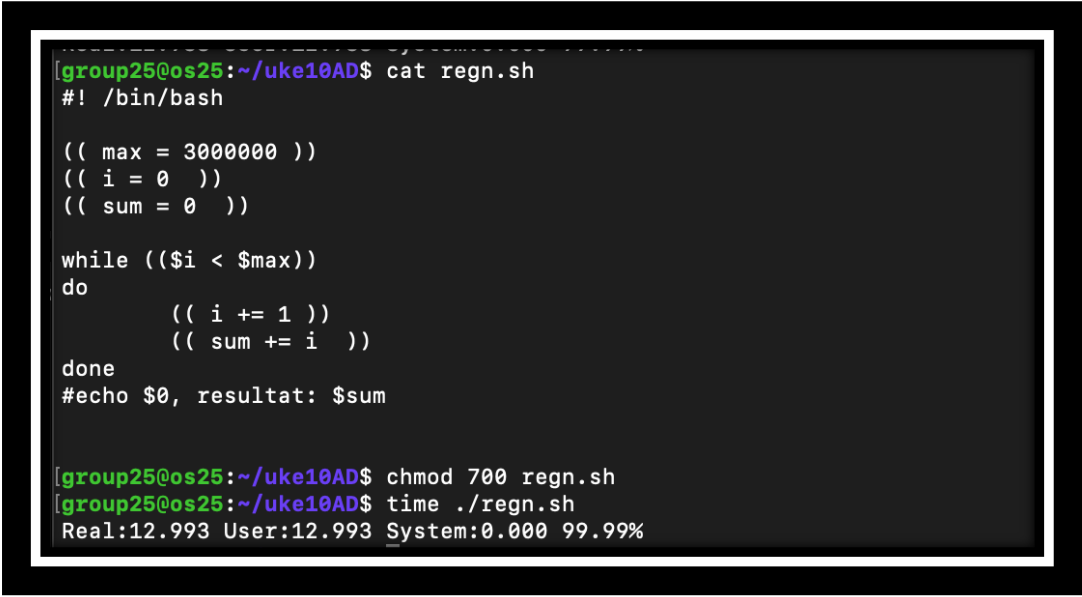
video- og bildebehandling, databehandling, og virtuelle maskiner. Det er viktig å merke seg at hyperthreading ikke gir en dobling av prosessorens ytelse, men kan gi en økning i effektiviteten til systemet hvis det implementeres riktig og brukes med riktig programvare og maskinvare.

En single core CPU kan inneholde to prosessorer samtidig: (operativsystemet opplever dette som to selvstendige prosessorer). Hardware switcher selv mellom de to prosessene i løpet av nanosekunder.

Det som faktisk skjer når operativsystemet fordeler prosesser til denne prosessoren som er hyperthreading, så settes det i gang to prosessorer på samme CPU, men de to prosessoren deler ALU-en som er på denne CPU-en.

### 3. (OBLIG)

Under skriver jeg inn koden, endrer mod og kjører med time for å sørge for at dette er riktig.



```
[group25@os25:~/uke10AD$ cat regn.sh
#!/bin/bash

(( max = 3000000 ))
(( i = 0 ))
(( sum = 0 ))

while (( $i < $max ))
do
    (( i += 1 ))
    (( sum += i ))
done
#echo $0, resultat: $sum

[group25@os25:~/uke10AD$ chmod 700 regn.sh
[group25@os25:~/uke10AD$ time ./regn.sh
Real:12.993 User:12.993 System:0.000 99.99%
```

Din VM er egentlig en docker-container som er startet med `docker container run --cpus="2"` slik at den bare får det som tilsvarer 2 CPU-er med CPU-tid. Prøv å finne ut av effekten av hyperthreading for CPUene på Linux-VM ved å samtidig kjøre regn-scriptet ved hjelp av taskset på to CPU-siblings og på to CPU-er som ikke er siblings og ta tiden på dem. Utgjør dette noen forskjell utifra tiden det tar? Hva kan du konkludere utifra dette?

Har kjørt med to CPU-siblings og to som ikke er siblings. Ettersom det er en betydelig forskjell i tiden det tar å kjøre, kan man konkludere med at hyperthreading har en effekt på CPU-ytelsen. Dette skyldes at to logiske kjerner i hyperthreading-teknologien deler ressurser på en fysisk kerne, og dermed kan det være en konkurranse om ressurser som fører til lavere ytelse når flere logiske kjerner kjører samtidig.

#### 4. (IKKE OBLIG)

Det er en forskjell i tiden det tar å kjøre koden dersom den kjøres med to CPU-siblings eller på to separate CPU-er fordi programmet er CPU-intensivt. Dette skyldes at CPU-siblings deler noen av ressursene som ALU, og kan derfor ikke utnytte disse ressursene like effektivt som separate CPU-er.

#### 5. (OBLIG)

Bytter fra Linux-VM group til root og sjekker om docker er oppe og runner:

```
group25@os25:~$ sudo su
[sudo] password for group25:
root@os25:/home/group25# cd
root@os25:~# service docker status
* Docker is running
root@os25:~#
```

**OPPGAVE 1:** Under kan man se andre gangen jeg kjørte «docker run hello-world» går det fortere fordi jeg allerede har lastet ned imaget.

```
root@os25:~# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent
   it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

root@os25:~#
```

**OPPGAVE 2:** Under kjører jeg kommando som henter alpine og lister alle docker image

```
[root@os25:~# docker run alpine ls -l
total 56
drwxr-xr-x    2 root    root    4096 Feb 10 16:45 bin
drwxr-xr-x    5 root    root    340 Mar 14 22:15 dev
drwxr-xr-x    1 root    root    4096 Mar 14 22:15 etc
drwxr-xr-x    2 root    root    4096 Feb 10 16:45 home
drwxr-xr-x    7 root    root    4096 Feb 10 16:45 lib
drwxr-xr-x    5 root    root    4096 Feb 10 16:45 media
drwxr-xr-x    2 root    root    4096 Feb 10 16:45 mnt
drwxr-xr-x    2 root    root    4096 Feb 10 16:45 opt
dr-xr-xr-x 5184 root    root      0 Mar 14 22:15 proc
drwx-----    2 root    root    4096 Feb 10 16:45 root
drwxr-xr-x    2 root    root    4096 Feb 10 16:45 run
drwxr-xr-x    2 root    root    4096 Feb 10 16:45/sbin
drwxr-xr-x    2 root    root    4096 Feb 10 16:45 srv
dr-xr-xr-x   13 nobody  nobody    0 Mar 14 22:15 sys
drwxrwxrwt    2 root    root    4096 Feb 10 16:45 tmp
drwxr-xr-x    7 root    root    4096 Feb 10 16:45 usr
drwxr-xr-x   12 root    root    4096 Feb 10 16:45 var
root@os25:~#
```

Det som egentlig skjer når vi kaller/kjører «run»:

1. docker klienten kontakter docker daemon
2. docker daemon lager containeren og runner en kommando i den containeren
3. docker daemon streamer output av kommandoer til docker klienten

```
[root@os25:~# docker run alpine echo "hello from alpine"
hello from alpine
root@os25:~#
```

Over ser vi at docker klienten kjørte echo kommandoen i alpine containeren og gikk ut av den. Under runner jeg kommandoen som ikke viser noe på skjermen. Fordi disse interaktive shellene vil gå ut etter at de har runnet hvilke som helst skript kommandoer, med mindre de runnes i et interaktivt terminal som er med eksempel i neste bilde:

```
[root@os25:~# docker run alpine /bin/sh
root@os25:~#
```

Under er jeg inne i container shellen og jeg prøver ut kommandoer som `ls -l` og `uname -a`:

```

root@os25:~# docker run -it alpine /bin/sh
/ # ls -l
total 56
drwxr-xr-x  2 root    root    4096 Feb 10 16:45 bin
drwxr-xr-x  5 root    root    360  Mar 14 22:21 dev
drwxr-xr-x  1 root    root    4096 Mar 14 22:21 etc
drwxr-xr-x  2 root    root    4096 Feb 10 16:45 home
drwxr-xr-x  7 root    root    4096 Feb 10 16:45 lib
drwxr-xr-x  5 root    root    4096 Feb 10 16:45 media
drwxr-xr-x  2 root    root    4096 Feb 10 16:45 mnt
drwxr-xr-x  2 root    root    4096 Feb 10 16:45 opt
dr-xr-xr-x 5173 root    root      0 Mar 14 22:21 proc
drwx----- 1 root    root    4096 Mar 14 22:21 root
drwxr-xr-x  2 root    root    4096 Feb 10 16:45 run
drwxr-xr-x  2 root    root    4096 Feb 10 16:45/sbin
drwxr-xr-x  2 root    root    4096 Feb 10 16:45 srv
dr-xr-xr-x 13 nobody nobody      0 Mar 14 22:21 sys
drwxrwxrwt  2 root    root    4096 Feb 10 16:45 tmp
drwxr-xr-x  7 root    root    4096 Feb 10 16:45 usr
drwxr-xr-x 12 root    root    4096 Feb 10 16:45 var
/ # uname -a
Linux 47dca5dd12f9 5.4.0-132-generic #148-Ubuntu SMP Mon Oct 17 16:02:06 UTC 2022 x86_64 Linux
/ # █

```

Bruker exit for å komme meg ut av shellet. Jeg tar docker ps for å vise alle containers som foreløpig kjører:

```

/ # exit
root@os25:~# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
432564db7c31   apacheubuntu   "/bin/bash"             32 hours ago  Up 32 hours   0.0.0.0:8089->80/tcp             stupefied_wright
435c4b86517b   apacheubuntu   "/bin/bash"             32 hours ago  Up 32 hours   0.0.0.0:8088->80/tcp             sleepy_wilson
2ab5f6b9a502   apacheubuntu   "/bin/bash"             32 hours ago  Up 32 hours   0.0.0.0:8087->80/tcp             stupefied_boyd
9979fc6403dc   apacheubuntu   "/bin/bash"             32 hours ago  Up 32 hours   0.0.0.0:8086->80/tcp             loving_aryabhata
4eb362f7f269   apacheubuntu   "/bin/bash"             32 hours ago  Up 32 hours   0.0.0.0:8085->80/tcp             gallant_dhawan
6080c88694fa   apacheubuntu   "/bin/bash"             32 hours ago  Up 32 hours   0.0.0.0:8084->80/tcp             flamboyant_hawking
a3cbc3e93529   apacheubuntu   "/bin/bash"             32 hours ago  Up 32 hours   0.0.0.0:8083->80/tcp             unruffled_bardeen
1f21b9721646   apacheubuntu   "/bin/bash"             32 hours ago  Up 32 hours   0.0.0.0:8082->80/tcp             modest_cray
e7715a419690   apacheubuntu   "/bin/bash"             32 hours ago  Up 32 hours   0.0.0.0:8081->80/tcp             jolly_burnell
root@os25:~# █

```

Med docker ps -a vises alle containere inkludert kjørende og

```

root@os25:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
47dca5dd12f9   alpine        "/bin/sh"               3 minutes ago  Exited (0)   About a minute ago              heuristic_dirac
722f435f0f6f   alpine        "/bin/sh"               4 minutes ago  Exited (0)   4 minutes ago                   dreamy_babbage
71dbfdf34dfd   alpine        "echo 'hello from al... 6 minutes ago  Exited (0)   6 minutes ago                   pedantic_wright
3f182485fa10   alpine        "ls -l"                 9 minutes ago  Exited (0)   9 minutes ago                   focused_ardinghelli
b46936e8ca0e   hello-world   "/hello"                16 minutes ago Exited (0)   16 minutes ago                  focused_vaughan
3a0036504e12   hello-world   "/hello"                19 minutes ago Exited (0)   19 minutes ago                  fervent_shtern

```

TIPS ER Å BRUKE --NAME OPSJONEN FOR Å KALLE CONTAINERE EGEN BESTEMTE NAVN!

**OPPGAVE 3:**

Kobler opp til containeren igjen:

```
See 'docker run --help'.
[root@os25:~# docker run -ti alpine
/ #
```

Kjører noen kommandoer

```
root@os25:~# docker run -ti alpine
/ # ls /
bin      dev      etc      home     lib      media    mnt      opt      proc     root     run      sbin     srv      sys
var
/ # whoami
root
```

Stopper containeren og sletter den. «docker image rm id» vil UNTAG-e imaget og fjerner referansene til «the sha of the image».

```
root@os25:~# docker stop f488c059ddc7
f488c059ddc7
root@os25:~# docker rm f488c059ddc7
f488c059ddc7
root@os25:~# docker container ls -a
```

Her kan jeg se helt til høyre at alpine tar en god del plass men selve containeren tar 0B.

```
root@os25:~# docker container ls -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES          SIZE
71dbdf34d9d   alpine     "echo 'hello from al_..." 24 minutes ago Exited (0) 23 minutes ago           pedantic_wright 0B (virtual 7.04MB)
3f182485fa10   alpine     "ls -l"                  27 minutes ago Exited (0) 26 minutes ago           focused_ardighelli 0B (virtual 7.04MB)
b46936e8ca8e   hello-world "/hello"                34 minutes ago Exited (0) 34 minutes ago           focused_vaughan 0B (virtual 13.3KB)
3a0836504e12   hello-world "/hello"                38 minutes ago Exited (0) 37 minutes ago           fervent_shtern 0B (virtual 13.3KB)
```

Det er også mulig å lage en container, men samtidig si at man ønsker at den skal slette containeren med engang den stopper ved f eks: «docker run --rm -it alpine».

Når man bygger, runner og rebygger image laster man ned veldig mange layers. Disse slettes ikke og docker leverer en kommando «PRUNE»:

- docker container prune
- docker image prune
- docker network prune
- docker volume prune



The docker image prune command allows you to clean up unused images. By default, docker image prune only cleans up dangling images. A dangling image is one that is not tagged and is not referenced by any container. To remove all *unused* resources, resources that are not directly used by any existing containers, use the `-a` switch as well.

If you want a general cleanup, then `docker system prune` is your friend.

#### OPPGAVE 4:

Kjører kommando `docker run -p 8080:80 nginx`

```
root@os25:~# docker run -p 8080:80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/03/15 13:33:01 [notice] 1#1: using the "epoll" event method
2023/03/15 13:33:01 [notice] 1#1: nginx/1.23.3
2023/03/15 13:33:01 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/03/15 13:33:01 [notice] 1#1: OS: Linux 5.4.0-132-generic
2023/03/15 13:33:01 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/03/15 13:33:01 [notice] 1#1: start worker processes
2023/03/15 13:33:01 [notice] 1#1: start worker process 82
2023/03/15 13:33:01 [notice] 1#1: start worker process 83
```

Under kan man se at jeg stopper og fjerner docker nginx:

```
root@os25:~# docker stop 16c923031e0c7ea12040f3bdafed0919b1bf5e94483b9016b50e17acb1a97e32
16c923031e0c7ea12040f3bdafed0919b1bf5e94483b9016b50e17acb1a97e32
root@os25:~# docker container ls -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
16c923031e0c	nginx	"/docker-entrypoint..."	56 seconds ago	Exited (0) 13 seconds ago
o	pedantic_mirzakhani			
f97fabac6c94	nginx	"/docker-entrypoint..."	5 minutes ago	Exited (0) About a minute ago
e	festive_bell			

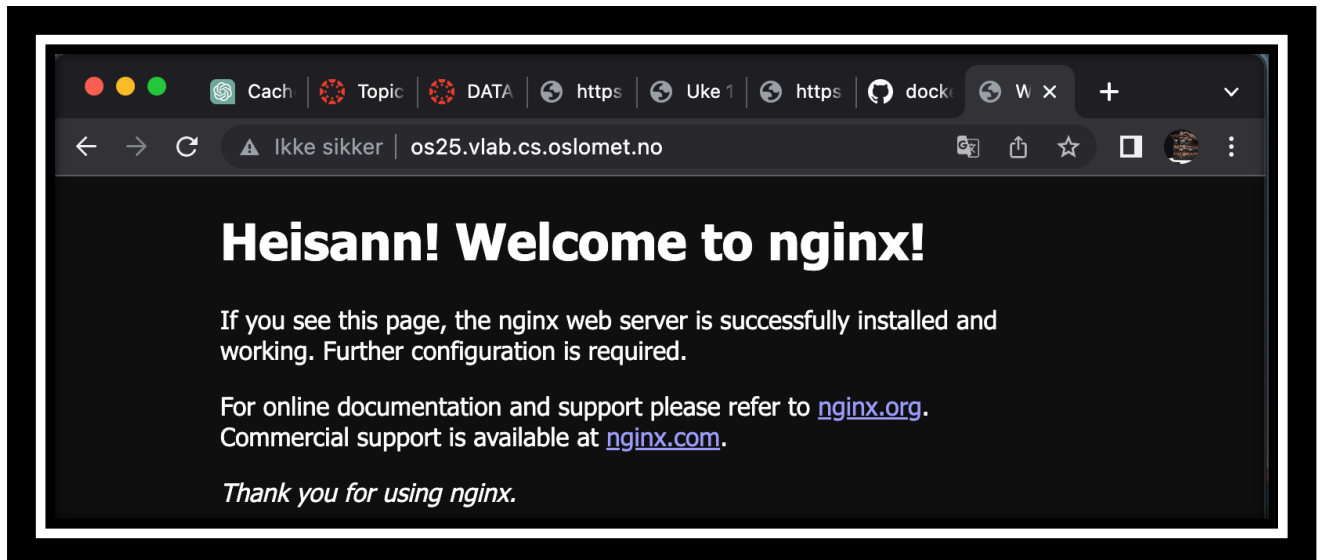
#### OPPGAVE 5:

Starter med å runne en nginx container

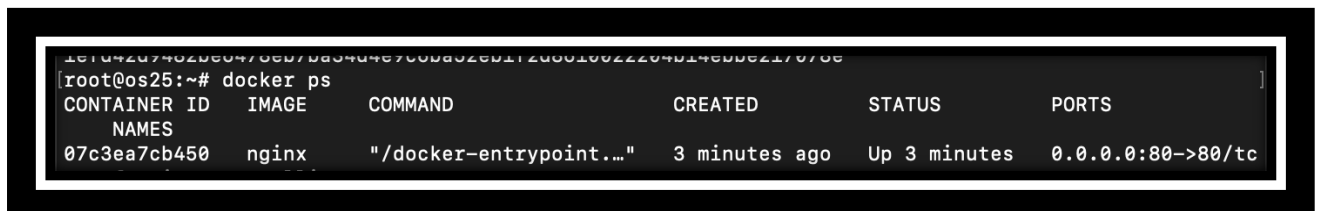
```
Error response from daemon: No such container: oppg5
root@os25:~# docker run -d -p 8000:80 nginx
1efd42d9482be6478eb7ba34d4e9c6ba52eb1f2d8610022204b14ebbe217078e
```

Under kan man se at den runner riktig:

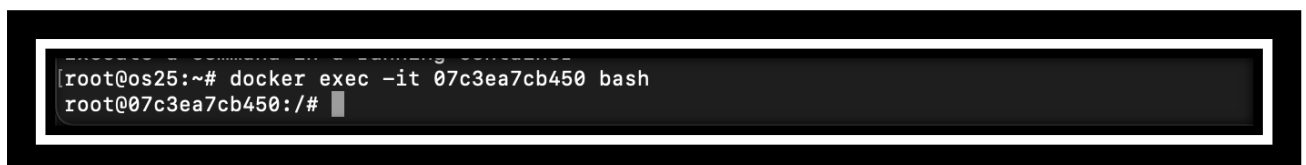




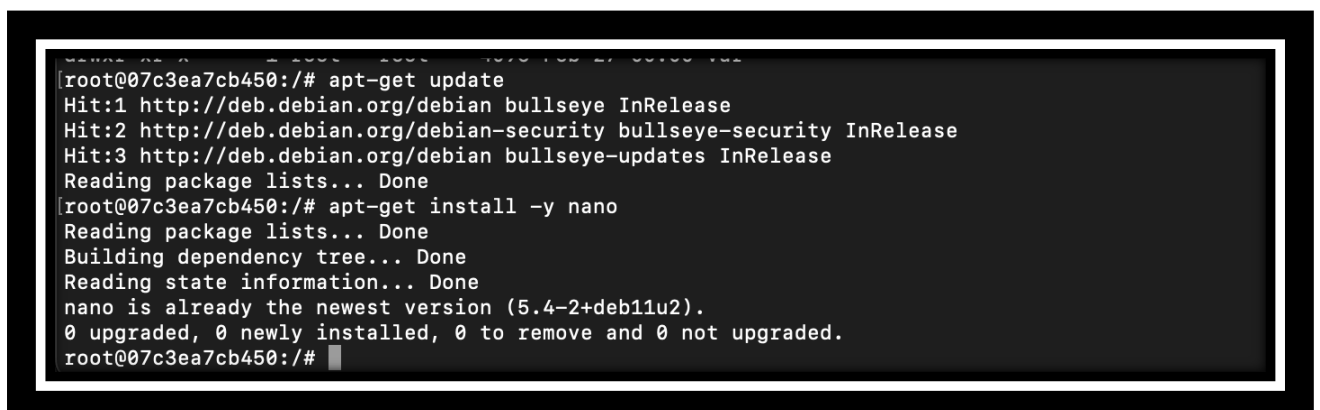
Sjekker navnet til containeren:



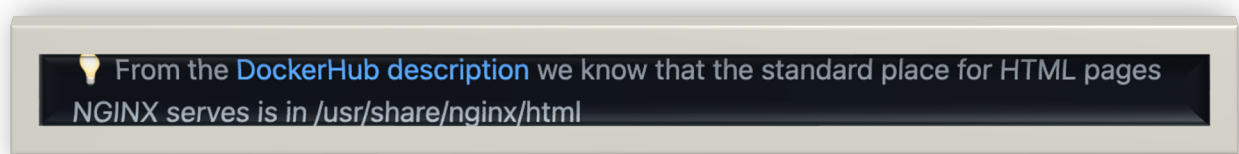
Stepper inn i en ny container av å execute et bash shell inni containeren som jeg startet:



Oppdaterer og laster ned nano for å redigere på HTML fila:



Vi vet at html ligger inni /usr/share/nginx/html



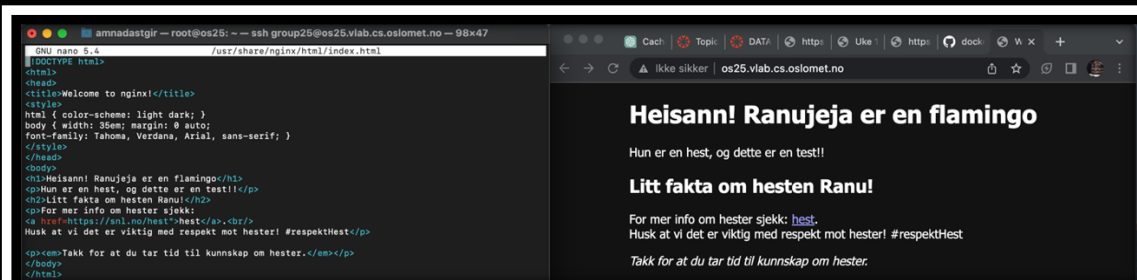
Her har jeg gjort nødvendige oppdatering og installasjoner for å hente nano tekstredigeringsverktøyet:

```

root@07c3ea7cb450:/# apt-get update
Hit:1 http://deb.debian.org/debian bullseye InRelease
Hit:2 http://deb.debian.org/debian-security bullseye-security InRelease
Hit:3 http://deb.debian.org/debian bullseye-updates InRelease
Reading package lists... Done
root@07c3ea7cb450:/# apt-get install -y nano
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nano is already the newest version (5.4-2+deb11u2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

```

Flere endringer på siden:



```

GNU nano 5.4 /usr/share/nginx/html/index.html
[ROGTYPE html]
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Heisann! Ranujeja er en flamingo</h1>
<p>Hun er en hest, og dette er en test!!</p>
<p>Litt fakta om hesten Ranu</p>
<p>For mer info om hester sjekk:
<a href="https://snl.no/hest">hest</a><br>
Husk at vi det er viktig med respekt mot hester! #respektHest</p>
<p>Takk for at du tar tid til kunnskap om hester.</p>
</body>
</html>

```

Heisann! Ranujeja er en flamingo

Hun er en hest, og dette er en test!!

Litt fakta om hesten Ranu!

For mer info om hester sjekk: [hest](https://snl.no/hest).  
Husk at vi det er viktig med respekt mot hester! #respektHest

Takk for at du tar tid til kunnskap om hester.

For å dra ut av containeren:

```

root@07c3ea7cb450:~# exit
exit
root@os25:~#

```

Stopper og fjerner docker containerne for å unngå mange lag som bygges oppå hverandre.

```

root@os25:~# docker stop 62d28dd30b2d
62d28dd30b2d
root@os25:~# docker rm 62d28dd30b2d
62d28dd30b2d
root@os25:~# docker stop 07c3ea7cb450
07c3ea7cb450
root@os25:~# docker rm 07c3ea7cb450
07c3ea7cb450

```

## 6. (OBLIG)

Starter en ubuntu container og inkluderer porten:

```

root@os25:~# docker run -dit -p 8081:80 ubuntu
4d296982a897473445d4a6db519e0f6b0bf6b9b82ba4fb554f764d02fe4ecc8c

```

Går inn i containeren med container id som jeg finner ved å ta «docker ps -a» for å se de kjørende containerne:

```
root@os25:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
4d296982a897   ubuntu        "/bin/bash"             About a minute Up About a minute   0.0.0.0:8081->80/tcp    upbeat_noether
63b793ff4929   apacheubuntu  "/bin/bash"             26 hours ago  Up 26 hours       0.0.0.0:7084->80/tcp    sharp_jepsen
21d1996f8f84   apacheubuntu  "/bin/bash"             26 hours ago  Created                                     gracious_agnesi
```

Går inn i containeren jeg lagde med hjelp av container id (eller hostname):

```
root@os25:~# docker exec -it 4d296982a897 bash
root@4d296982a897:/#
```

Tar «apt-get update» og «apt-get install apache2» inne i containeren:

```
root@4d296982a897:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [23.2 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [868 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [900 kB]
```

```
root@4d296982a897:/# apt-get install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils bzip2 ca-certificates file libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libbrotli1
  libcurl4 libexpat1 libgdbm-compat4 libgdbm6 libicu70 libjansson4 libldap-2.5-0 libldap-common liblua5.3-0 libmagic-mgc libmagic1

```

Navigerer meg fram til mappa der index.html fila ligger:

```
root@4d296982a897:/# cd /var/www/html/
root@4d296982a897:/var/www/html# ls -l
total 12
-rw-r--r-- 1 root root 10671 Mar 16 16:48 index.html
```

Sender inn en melding til index.html fila:

```
root@4d296982a897:/var/www/html# echo "Docker image fra Amna, tester om siden fungerer" > index.html
root@4d296982a897:/var/www/html# cat index.html
Docker image fra Amna, tester om siden fungerer
```

Kan se at alt fungerer:

```
root@4d296982a897:/var/www/html# ls -l
total 4
-rw-r--r-- 1 root root 48 Mar 16 17:10 index.html
root@4d296982a897:/var/www/html# cat index.html
Docker image fra Amna, tester om siden fungerer
```

Går ikke (skjønner ikke hvorfor, er koblet til skolenett)

```
root@4d296982a897:/var/www/html# curl http://localhost
curl: (7) Failed to connect to localhost port 80 after 0 ms: Connection refused
root@4d296982a897:/var/www/html# curl http://127.0.0.1
curl: (7) Failed to connect to 127.0.0.1 port 80 after 0 ms: Connection refused
root@4d296982a897:/var/www/html#
```

Lister alle containere:

```
root@os25:~# docker container ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
4d296982a897   ubuntu        "/bin/bash"             32 minutes ago Up 32 minutes 0.0.0.0:8081->80/tcp               upbeat_noether
63b793ff4929   apacheubuntu  "/bin/bash"             26 hours ago  Up 26 hours  0.0.0.0:7084->80/tcp               sharp_jepsen
21d1996f8f84   apacheubuntu  "/bin/bash"             26 hours ago  Created                                gracious_agnes
dbf8f260ef1a   apacheubuntu  "/bin/bash"             26 hours ago  Created                                jovial_diffie
8713ee4c2540   apacheubuntu  "/bin/bash"             26 hours ago  Created                                wonderful_jemison
ab9a3e69dbdd   apacheubuntu  "/bin/bash"             26 hours ago  Up 26 hours  0.0.0.0:7083->80/tcp               nice_leakey
7aaf636731c0   apacheubuntu  "/bin/bash"             26 hours ago  Up 26 hours  0.0.0.0:7082->80/tcp               flamboyant_chatterjee
458751d33e5e   apacheubuntu  "/bin/bash"             26 hours ago  Up 26 hours  0.0.0.0:7081->80/tcp               agitated_taussig
40b4aff49021   apacheubuntu  "/bin/bash"             27 hours ago  Up 27 hours  0.0.0.0:7072->80/tcp               objective_taussig
9478035499d9   ubuntu        "/bin/bash"             27 hours ago  Up 27 hours  0.0.0.0:7071->80/tcp               busy_babbage
```

Lager en kopi av containeren:

```
root@os25:~# docker container commit 4d296982a897 apacheubuntu
sha256:77db188e41b0ef49dcdf8691ed4d89129cae28beb1d79a226330f6970efcb579
```

Starter en ny container med apache installert ved:

```
root@os25:~# docker container run -it -d -p 8082:80 apacheubuntu /bin/bash
ce5d0e49e0dd3052e04e76a3e065ec2618711e60c318c7fb369a4ed858d02128
```

Lister docker for å få id til nye containeren

```
root@os25:~# docker container ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
ce5d0e49e0dd   apacheubuntu  "/bin/bash"             5 minutes ago Up 5 minutes  0.0.0.0:8082->80/tcp               hopeful_kare
4d296982a897   ubuntu        "/bin/bash"             47 minutes ago Up 47 minutes  0.0.0.0:8081->80/tcp               upbeat_noether
63b793ff4929   28dd882c28e2  "/bin/bash"             27 hours ago  Up 27 hours  0.0.0.0:7084->80/tcp               sharp_jepsen
```

starte opp en ny container med apache installert ved:

```
root@os25:~# docker container exec ce5d0e49e0dd /bin/bash -c "/etc/init.d/apache2 start"
* Starting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.9. Set the 'ServerName' directive globally to suppress this message
*
root@os25:~#
```

Logger meg inn i den nye containeren som jeg lagde ved å kopiere den gamle:

```

root@os25:~# docker exec -it ce5d0e49e0dd bash
root@ce5d0e49e0dd:/# cd /var/www/html
root@ce5d0e49e0dd:/var/www/html# ls -l
total 4
-rw-r--r-- 1 root root 48 Mar 16 17:10 index.html

```

Endrer inni fila til denne html for å kunne merke forskjell på instansene:

```

root@ce5d0e49e0dd:/var/www/html# echo "dette er testversjonen, som vil vise forskjellen med containeren jeg kopierte fra" > index.html
root@ce5d0e49e0dd:/var/www/html# cat index.html
dette er testversjonen, som vil vise forskjellen med containeren jeg kopierte fra

```

## KLARTE Å FÅ TIL CURL!

```

root@ce5d0e49e0dd:/var/www/html# curl http://localhost
dette er testversjonen, som vil vise forskjellen med containeren jeg kopierte fra

```

## 7. UKENS UTFORDRING!

Sliter som bare det, men her er koden min:

```

group25@os25:~/uke10AD$ cat oppg7
#!/bin/bash

for port in {8081..8089}
do
    containerID=$(docker run -d -p $port:80 httpd:latest)
    echo "<html><body><h1>Web server running on port $port</h1></body></html>" | docker exec -i $containerID sh -c 'cat > /usr/local/apache2/htdocs/index.html'
    echo "Container $containerID started on port $port"
done
group25@os25:~/uke10AD$

```

```

File key ==> File Edit Search Buffers Windows System Help
#!/bin/bash

for port in {8081..8089}
do
    containerID=$(docker run -d -p $port:80 httpd:latest)
    echo "<html><body><h1>Web server running on port $port</h1></body></html>" | docker exec -i $containerID sh -c 'cat > /usr/local/apache2/htdocs/index.html'
    echo "Container $containerID started on port $port"
done

```

Fungerer:

```

group25@os25:~/uke10AD$ jed oppg7
Container 3bbaf7d0c719ab09551b504232051d207886240e808461895cd54164def69edd started on port 8082
Container aeb2b35e1664bbd6a1fde92a599c2092cac1fafa283b7bb704b38aa5a002330 started on port 8083
Container f7eb1f3e88ccbd3c1a56311285b917c833c943a8223ffdfdf7f784104bdc8c started on port 8084
Container 410ea397d8c7482af03758981e9a7ce3649e9cbdd76b3888b52d0a9f46d47dfc started on port 8085
Container 4c2ce817f2fbbcc9415049d8ae6d88cec980de9d82693ea937b6f7d6f78a1f21d started on port 8086
Container c5603aaf1674d7faae90b2910f826cfcca292b08bc482d8a6ae9e19fb12e60d6 started on port 8087
Container 38f24b8e4e327312e44a71b51b4f496d6f0a85862135e45c6b90b843bbdf6ec8 started on port 8088
Container 465708c776097492358757dee44873990fa4e3f2eb0d9b4ec2ba6c6bc88b5e03 started on port 8089
group25@os25:~/uke10AD$

```

**8. (IKKE OBLIG)**

Den vesentlige forskjellen mellom å kjøre 10 docker-instanser med hver sin Apache webserver og kjøre 10 virtuelle maskiner med hver sin Apache webserver: er ressursbanken.

Når man kjører virtuelle maskiner, vil hver virtuell maskin ha sin egen fullstendige kopiering av operativsystemet og all programvare som kreves for å kjøre en Apache webserver. Dette vil føre til at hver virtuell maskin vil bruke betydelige mengder av RAM og diskplass for å kjøre. Dermed kan du ende opp med å bruke mye mer RAM og diskplass enn det som er nødvendig.

Når du kjører docker-instanser, deler hver instans samme operativsystemkjernen og annen programvare med verts-OS-et. Dette fører til at hver docker-instans bruker mindre RAM og diskplass enn det som kreves for å kjøre en virtuell maskin. Dermed kan du kjøre flere docker-instanser på samme server sammenliknet med virtuelle maskiner, uten at det går på bekostningen av ressursbanken.

Så, hvis man ønsker å kjøre flere instanser av en Apache webserver på samme server uten å bruke mye ressurser, ville det være mer effektivt å bruke docker-instanser enn virtuelle maskiner.

