

Oppgave 1

Hvor mange prosesser kjører på din datamaskin?

Hvor mange prosesser som kjøres på en datamaskin endres hele tiden, og dette avhenger av hvilke, og hvor mange programmer man har åpnet. For øyeblikket kjøres følgende:

Windows-prosesser: 31

Bakgrunnsprosesser : 71

"Apper" : 5

Totalt: 107

Nikolai	Marius	Benjamin	Sindre(mac)	Shiwan	Ella	Gisle(mac)	Erik
136	107	110	216	158	92	243	90

Interessant å se at Mac nesten har det dobbelte av hva Windows har.

Hvor mange prosesser som kjører på din virtuelle server i nettskyen?

For å sjekke prosesser under linux/i bash kan man bruke kommandoen "top"

Denne resulterte i at det for øyeblikket var 124 kjørende prosesser, 1 som var i bruk, og 123 som var i "sleep", altså venter på å bli brukt.

```
top - 14:41:58 up 1:46, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 124 total, 1 running, 123 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4046588 total, 3492092 free, 48936 used, 505560 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 3751208 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	38072	5976	3836	S	0.0	0.1	0:01.35	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:+
6	root	20	0	0	0	0	S	0.0	0.0	0:00.09	kworker/u4+
7	root	20	0	0	0	0	S	0.0	0.0	0:00.06	rcu_sched
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	watchdog/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	watchdog/1
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/1
13	root	20	0	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/1
15	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/1:+
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
17	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	netns
18	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	perf
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd

Nikolai	Marius	Benjamin	Sindre(mac)	Shiwan	Ella	Gisle(mac)	Erik
124	124	125	118	118	117	118	118

Kan man gi et nøyaktig antall? Begrunn.

Man kan gi et nøyaktig antall på et punkt, men dette vil hele tiden variere, og jeg kan ikke for eksempel si at om 5 minutter er det 130 prosesser som kjører, for dette kan jeg ikke vite på forhånd.

Hvor mange av prosessene som "kjører"?

For øyeblikket så er det bare 1 av prosessene som virkelig "kjører", de resterende venter på inputs for at de skal bli tatt i bruk.

Hvis de ikke kjører, hvilke tilstander befinner de seg da?

Hvis prosessene ikke kjører ligger de i en "sleep" tilstand, altså de venter på inputs. Uten input så kan ikke prosessene gjøre noe, så de må bare vente helt til vi gir en input som en av de kan forstå.

Hva er maskinvarespesifikasjon til din datamaskin (noter prosessortype, prosessorarkitektur, klokkefrekvens, informasjon om primært minne, størrelse på cache (både L1, L2 og L3 er ønskelig))?

Prosessortype: Intel core i5 4200M

Arkitektur: Intel Haswell, som er basert på x64 arkitekturen

Klokkefrekvens: 2500GHz (Basis) 3100GHz (med "turbo mode", automatisk overklokking"

Primært minne: Hvis primært minne er RAM så 8192MB

Cache:

L1 Data Cache Size	2 x 32 KBytes
L1 Instructions Cache Size	2 x 32 KBytes
L2 Unified Cache Size	2 x 256 KBytes
L3 Unified Cache Size	3072 KBytes

Nikolai	Marius	Benjamin	Sindre(mac)
AMD K16	Det som står over	Intel Core i5-2467	Intel Core i5-4260U,
Prosesorarkitektur Mullins		Arkitektur: Sandy Bridge	Prosesorarkitektur Q2,
Klokke frekvens 2.20 Gz		Frekvens: 1.6GHz	Klokkefrekvens 1.6GHZ
L1 256 kB		L1, 2x32kB x2	RAM - 8GB,
L2 2,0 kB		L2, 2x256kB	L2 - 256kb
		L3, 3072kB	L3 - 3MB
		Ram: 4,00GB DDR3	
		665MHz	

Shiwan	Gisle	Erik	Ella
--------	-------	------	------

intel(R) core (TM) i5-6200	Processor: 1,3 GHz Intel Core i5)	Intel(R) Core™ i7-6500U	Intel Core i7 5500U
Arkitektur: skylake-U/Y x64 basert prosessor	Minne: 4GB	Skylake arkitektur	Arkitektur: Broadwell-U
Frekvens 2.40ghz	Prosesorarkitektur 64bit	klokke frekvens 2.5GHz	Klokkefrekvens: 2.40GHz
L1: 128 KBYTES	Grafikk: Intel HD Graphics 5000 1536 MB	L1 : 128kB	L1: 128 KB
L2: 512KBYTES		L2 : 512kB	L2: 512 KB
L3: 3072 MBYTES		L3 : 4MB	L3: 4.0 MB
		Ram : 8GB	

Hvor mange CPU-“cores” har du tilgjengelig på din maskin? Noter.

Nikolai	Marius	Benjamin	Sindre(mac)	Shiwan	Ella	Gisle(mac)	Erik
4	2	2	2	2	2	2	2

Hvor mange CPU-“cores” har du tilgjengelig på din virtuelle server? Noter.

På min virtuelle server valgte vi medium "Flavor", som vil si at jeg fikk 2 vCPUer.

Specs

Flavor Name	m1.medium
Flavor ID	47d7f445-db26-4f1d-bf58-e79de7394f97
RAM	4GB
VCPUs	2 VCPU
Disk	20GB

Dette er kanskje litt overkill i forhold til hva jeg egentlig behøver, men den gir oss maks total kapasitet inne i UH-IAAS.

Finn ut hvilken prosess i ditt system bruker mest minne. Beskrive denne prosessen

Kort

På min maskin er det for øyeblikket Google Chrome som bruker mest minne.

Google chrome er en nettleter, og den er strukturert slik at hver enkelt fane er en egen prosess.

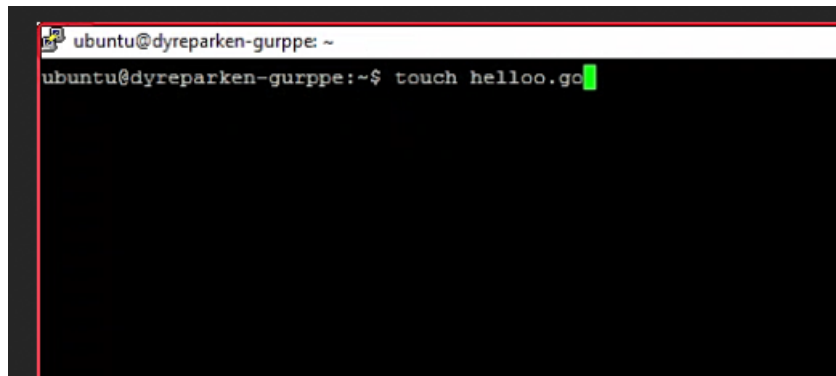
Dersom den ene skulle kræsje, så påvirker den ikke de andre fanene slik at de enkelt kan leve videre for seg selv.

Nikolai	Marius	Benjamin	Sindre(mac)
---------	--------	----------	-------------

Google Chrome	På min maskin er det for øyeblikket Google Chrome som bruker mest minne. Google chrome er en nettleser, og den er strukturert slik at hver enkelt fane er en egen prosess. Dersom den ene skulle kræsje, så påvirker den ikke de andre fanene slik at de enkelt kan leve videre for seg selv.	Google Chrome starter nye prosesser for hver faner. Ved mange faner vil den kjøre mange prosesser.	Spotify-helper Finner ikke noe informasjon på hva denne prosessen gjør.
---------------	---	--	--

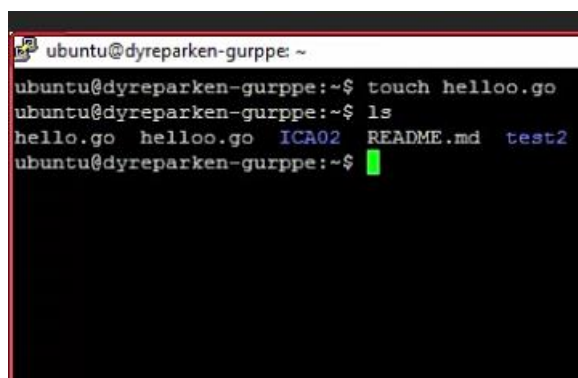
Erik	Ella	Shiwan	Gisle
Google Chrome Starter side med Chrome, ellers brukes det til alt mulig.	Google Chrome Starter side med Chrome eller brukes til alt mulig.	Opera Internet Browser Brukes til Streaming osv.	Spotify-helper Finner ikke noe informasjon på hva denne prosessen gjør.

Oppgave 2



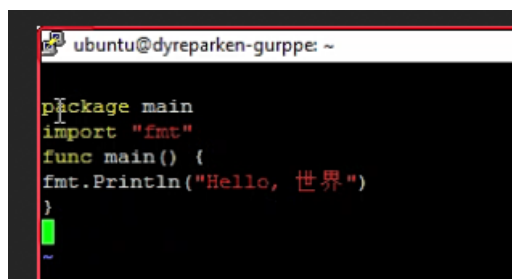
```
ubuntu@dyreparken-gurppe: ~  
ubuntu@dyreparken-gurppe:~$ touch helloo.go
```

1. Her lager vi en ny fil i virtuelle serveren ved å skrive touch helloo.go
Touch = kommando Helloo.go = navnet på filen



```
ubuntu@dyreparken-gurppe: ~  
ubuntu@dyreparken-gurppe:~$ touch helloo.go  
ubuntu@dyreparken-gurppe:~$ ls  
hello.go helloo.go ICA02 README.md test2  
ubuntu@dyreparken-gurppe:~$
```

2. Vi skriver kommandoen ls for å finne ut om filen er opprettet, og som vi ser så ligger det en ny fil med navnet helloo.go



```
ubuntu@dyreparken-gurppe: ~  
package main  
import "fmt"  
func main() {  
    fmt.Println("Hello, 世界")  
}
```

3. Da skriver vi kommandoen vi helloo.go for å kunne redigere filen og legge inn informasjonen vi fikk på pdf filen. Når det er skrevet inn, trykker vi på Shift + Z for å lagre.

```
ubuntu@dyreparken-gurppe: ~  
ubuntu@dyreparken-gurppe:~$ touch helloo.go  
ubuntu@dyreparken-gurppe:~$ ls  
hello.go  helloo.go  ICA02  README.md  test2  
ubuntu@dyreparken-gurppe:~$ vim helloo.go  
ubuntu@dyreparken-gurppe:~$
```

4. Vi skriver kommandoen ls for å få oversikt over filene vi har i serveren.

```
ubuntu@dyreparken-gurppe: ~  
ubuntu@dyreparken-gurppe:~$ touch helloo.go  
ubuntu@dyreparken-gurppe:~$ ls  
hello.go  helloo.go  ICA02  README.md  test2  
ubuntu@dyreparken-gurppe:~$ vim helloo.go  
ubuntu@dyreparken-gurppe:~$ ls  
hello.go  helloo.go  ICA02  README.md  test2  
ubuntu@dyreparken-gurppe:~$ GOOS=darwin GOARCH=386 go build helloo.go  
ubuntu@dyreparken-gurppe:~$
```

5. "på tvers" kompilering i golang gjør vi ved å spesifisere operativsystemtype og arkitektur foran "go build" kommandoet som vi fikk til delt. Denne filen er laget på windows masking og vi ønsker å akkserere den på en mac, derfor skriver vi inn den kommandoen :

GOOS=darwin GOARCH=386 go build helloo.go

```
ubuntu@dyreparken-gurppe: ~  
ubuntu@dyreparken-gurppe:~$ touch helloo.go  
ubuntu@dyreparken-gurppe:~$ ls  
hello.go  helloo.go  ICA02  README.md  test2  
ubuntu@dyreparken-gurppe:~$ vim helloo.go  
ubuntu@dyreparken-gurppe:~$ ls  
hello.go  helloo.go  ICA02  README.md  test2  
ubuntu@dyreparken-gurppe:~$ GOOS=darwin GOARCH=386 go build helloo.go  
ubuntu@dyreparken-gurppe:~$ ls  
hello.go  helloo  helloo.go  ICA02  README.md  test2  
ubuntu@dyreparken-gurppe:~$
```

6. vi bruker LS for å sjekke at filen helloo.go kompilert.

```
ubuntu@dyreparken-gurppe: ~  
ubuntu@dyreparken-gurppe:~$ git clone https://github.com/shiwanh/test3.git
```

7. vi cloner git hub repository.

```
ubuntu@dyreparken-gurppe: ~  
ubuntu@dyreparken-gurppe:~$ git clone https://github.com/shiwanh/test3.git  
Cloning into 'test3'...  
remote: Counting objects: 3, done.  
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
Unpacking objects: 100% (3/3), done.  
Checking connectivity... done.  
ubuntu@dyreparken-gurppe:~$ ls  
hello.go  helloo  helloo.go  ICA02  README.md  test2  test3  
ubuntu@dyreparken-gurppe:~$
```

8. ls for å sjekke at repositoryen “test3” er clonet

```
ubuntu@dyreparken-gurppe: ~  
ubuntu@dyreparken-gurppe:~$ mv helloo test3  
ubuntu@dyreparken-gurppe:~$
```

9. vi flytter filen helloo in til test3 (repository).

```
ubuntu@dyreparken-gurppe: ~/test3  
ubuntu@dyreparken-gurppe:~/test3$ ls  
helloo  README.md  
ubuntu@dyreparken-gurppe:~/test3$
```

10. vi skriver ls for å sjekke om filen ligger I repository

```
ubuntu@dyreparken-gurppe: ~/test3
ubuntu@dyreparken-gurppe:~/test3$ git checkout -b hellobranch
```

11. Her lager vi en ny branch som heter hellobranch

```
ubuntu@dyreparken-gurppe: ~/test3
ubuntu@dyreparken-gurppe:~/test3$ git add hello
ubuntu@dyreparken-gurppe:~/test3$ git status
On branch hellobranch
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   hello

ubuntu@dyreparken-gurppe:~/test3$ git commit -m "hello fil"
[hellobranch 7ffb13f] hello fil
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100755 hello
ubuntu@dyreparken-gurppe:~/test3$
```

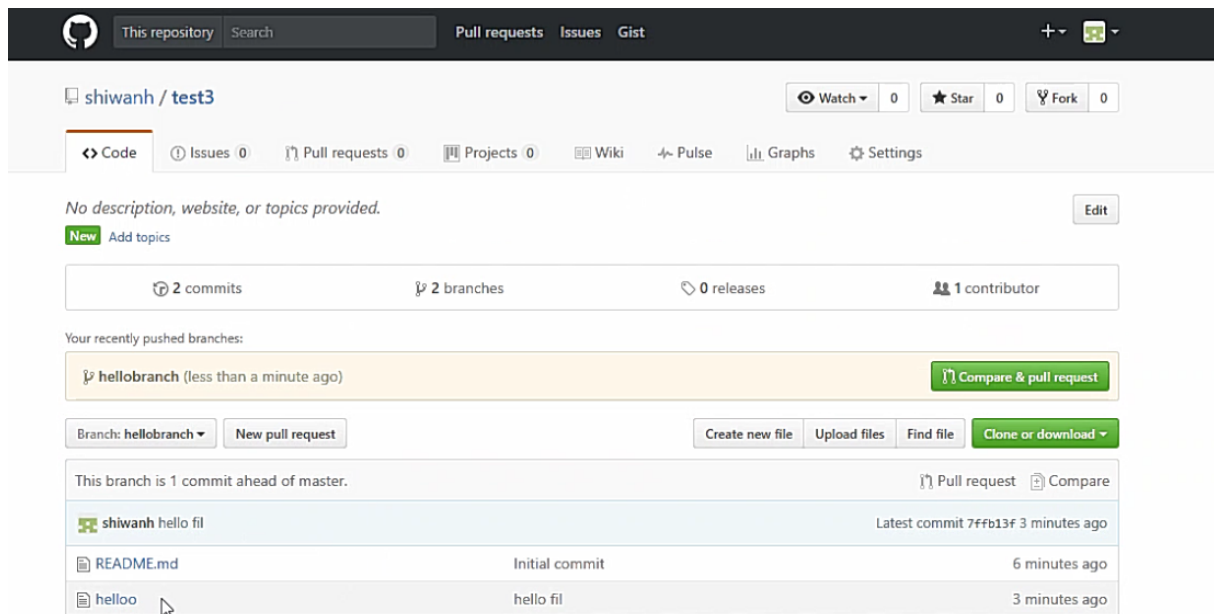
12. vi adder hello filen i repository ved å skrive git add hello

13. vi skriver git status for å sjekke statusen

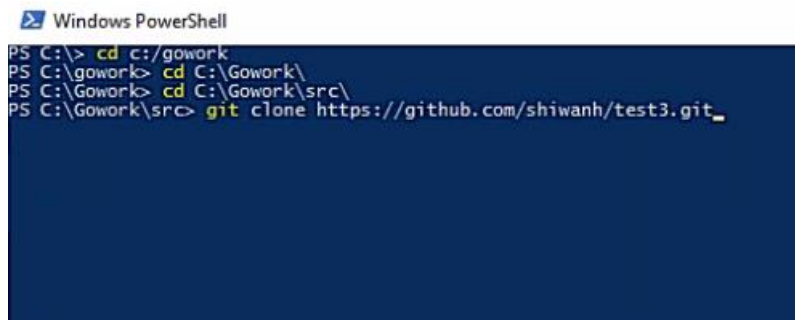
14. vi commiter filen ved å skrive git commit -m "hello fil"

```
ubuntu@dyreparken-gurppe:~/test3$ git push --set-upstream origin hellobranch
Username for 'https://github.com': shiwanh
Password for 'https://shiwanh@github.com':
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 644.17 KiB | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
```

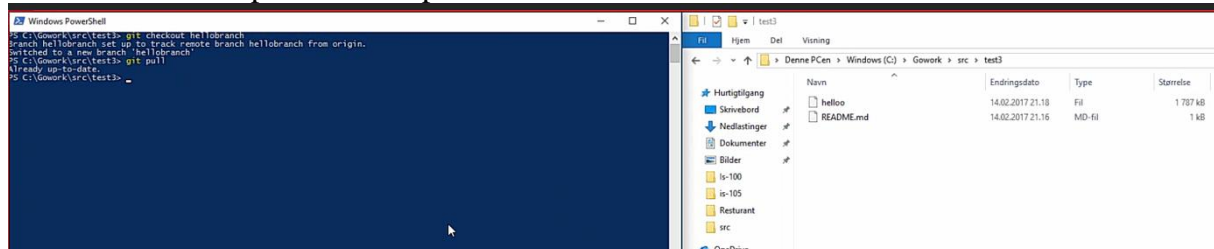
15. vi pusher filen ved å skrive git push --set-upstream origin hellobranch og da er filen lagret i repositoryen og er klar til henting.



16. her sjekker vi at filen er lastet opp I repositorien.



17. her cloner vi repositorien til pcen.



18. nå kan vi ha tilgang til filen på vår pc/mac.