Lektionstillfälle 14

Backup och arkivering
med Mikael Larsson

NACKADEMIN

Återblick

Förra lektionen arbetade vi med systemstart – med bootstrapping, systemd och crontab.

Dagens lektion

Mål: Att arbeta med filarkiv, göra backup lokalt och till molntjänst

- tar, gzip, zip
- rsync
- rclone

Termer och begrepp

- tar, att ”tara”
- gzip, att ”gzippa”
- bzip2
- komprimeringsalgoritmer
- rsync-skript

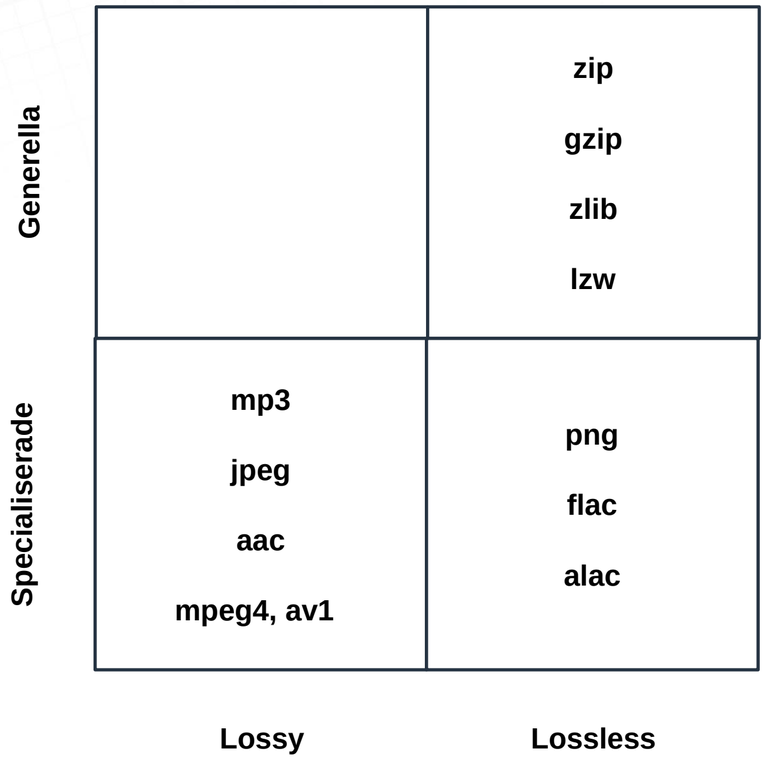
Datakomprimering

Ni har alla stött på och jobbat med komprimerade filer av olika slag i form av tar, gz, zip – filer, mp3:or, etc.

Ni använder också datakomprimering varje dag när ni tittar på en bild på er telefon, eller strömmar ljud eller video.

? Vilken är den stora skillnaden mellan den komprimering som zip gör och till exempel komprimeringen i en jpeg-bild?

Familjer av komprimeringsalgoritmer



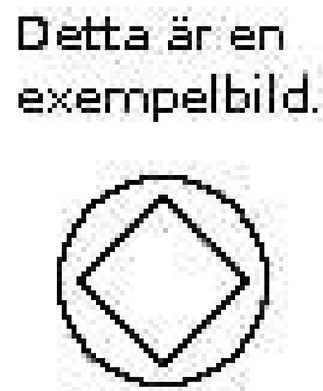
Lossy
Lossy betyder att du **inte kan** vara säker på att det du får tillbaka är identiskt med det du komprimerade.

Lossless
Lossless betyder att du **kan** vara säker på att det du får tillbaka är identiskt med det du komprimerade.

BMP - PNG - JPEG50 - JPEG10

29706 exempelbild.bmp
exempelbild.bmp: PC bitmap, Windows 98/2000 and newer format, 77 x 96 x 32, cbSize 29706, bits offset 138

1263 exempelbild.png
exempelbild.png: PNG image data, 77 x 96, 8-bit/color RGBA, non-interlaced



2261 exempelbild.50.jpeg
exempelbild.50.jpeg: JPEG image data, JFIF standard 1.01, resolution (DPI), density 300x300, segment length 16, baseline, precision 8, 77x96, components 3

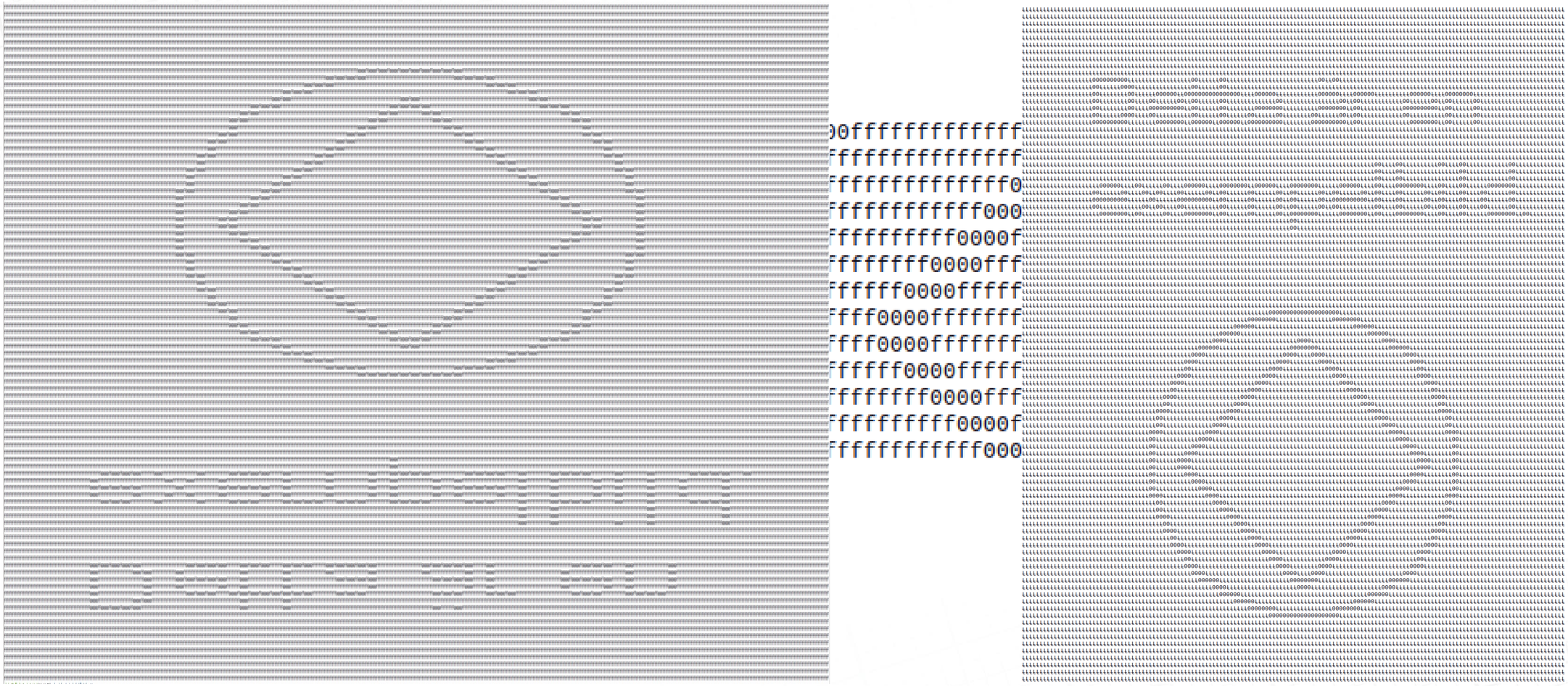
1938 exempelbild.10.jpeg
exempelbild.10.jpeg: JPEG image data, JFIF standard 1.01, resolution (DPI), density 300x300, segment length 16, baseline, precision 8, 77x96, components 3

Vanlig gzip komprimerar denna BMP till: 624 exempelbild.bmp.gz

NACKADEMIN

JPEG degradering efter 50 omsparningar

BMP formatet inuti



NACKADEMIN

Hur fungerar datakomprimering?

RLE, Run-length encoding
https://en.wikipedia.org/wiki/Run-length_encoding

Exempel: WWWWWWWWWWWWBWWWWWWWWWWWWBBBWWWWWWWWWWWWWWWWWWWWWW
Blir: 12W1B12W3B20W

Statistisk kodning
Exempel: Huffman
https://en.wikipedia.org/wiki/Huffman_coding

Olika slags block/fönster-komprimeringar
Exempel: LZ77

(+ massa andra varianter)

gzippad BMP-bild
I början på bilden är det ca 3000 **FF**
Detta kan komprimeras hårt.
Därför blir den gzippade bilden bara 624 tecken stor.

Komprimera filer

För att komprimera filer i Linux används främst gzip och bzip2.

De fungerar på liknande sätt, man anger en fil att komprimera,
eller så komprimerar de från **stdin**.

De kan alltså komprimera en ström.

```
$ gzip lotsofdata.txt
```

```
$ gzip < lotsofdata.txt > lotsofdata.txt.gz
```

”gzip” är snabbare, ”bzip2” komprimerar oftast bättre.
bzippad bildfil blev 466 bytes.

Observera att det bara är **en** fil som komprimeras.

Arkiv i Linux

Att packa ihop flera filer i en kallas att arkivera. Det har egentligen ingenting med ”arkivera” i vanlig bemärkelse att göra.

Man kan tänkta sig ett arkiv som att det innehåller många filer.

I Linux används ”tar” för att arbeta med arkiv.

”tar” = ”Tape ARchive” – ett mycket gammalt kommando.

En ren tar-fil är ***inte*** komprimerad.

”tar” – filer sparar filattribut på ett sätt som inte ”zip” kan. Det är därför alltid bättre att använda ”tar”.

tar - syntax

tar -[cxt]f tarfil.tar filer...

c = Create
x = eXtract
t = Test
f = File

Andra vanliga flaggor:
v = Verbose
z = stream is gZip-compressed
j = stream is bzip2-compressed
a = Automatically compress according to file suffix
C = Change dir before operation

”tar” arkiverar filträd rekursivt.

”tar” kan köras med massor av olika flaggor och alternativ. ”man”-bladet är över 1000 rader.

Komprimera arkiv

Följande genererar samma resultat:

```
$ tar -cf tarfil.tar filer && gzip tarfil.tar

$ tar -czf tarfil.tar.gz filer

$ tar -caf tarfil.tar.gz filer
```

Komprimera en ström

Som med de flesta kommandon som tar fil-argument kan man ange ”-” som filnamn för att läsa från **stdin** eller skriva till **stdout**.

Exemplet på föregående bild kan också utföras såhär:

```
$ tar -cf - filer | gzip > tarfil.tar.gz
```

På liknande sätt kan kan också kopiera hela filträd och komprimera datat ”i sladden”:

```
$ tar -C src -czf - filer | tar -C dst -xzf -
```

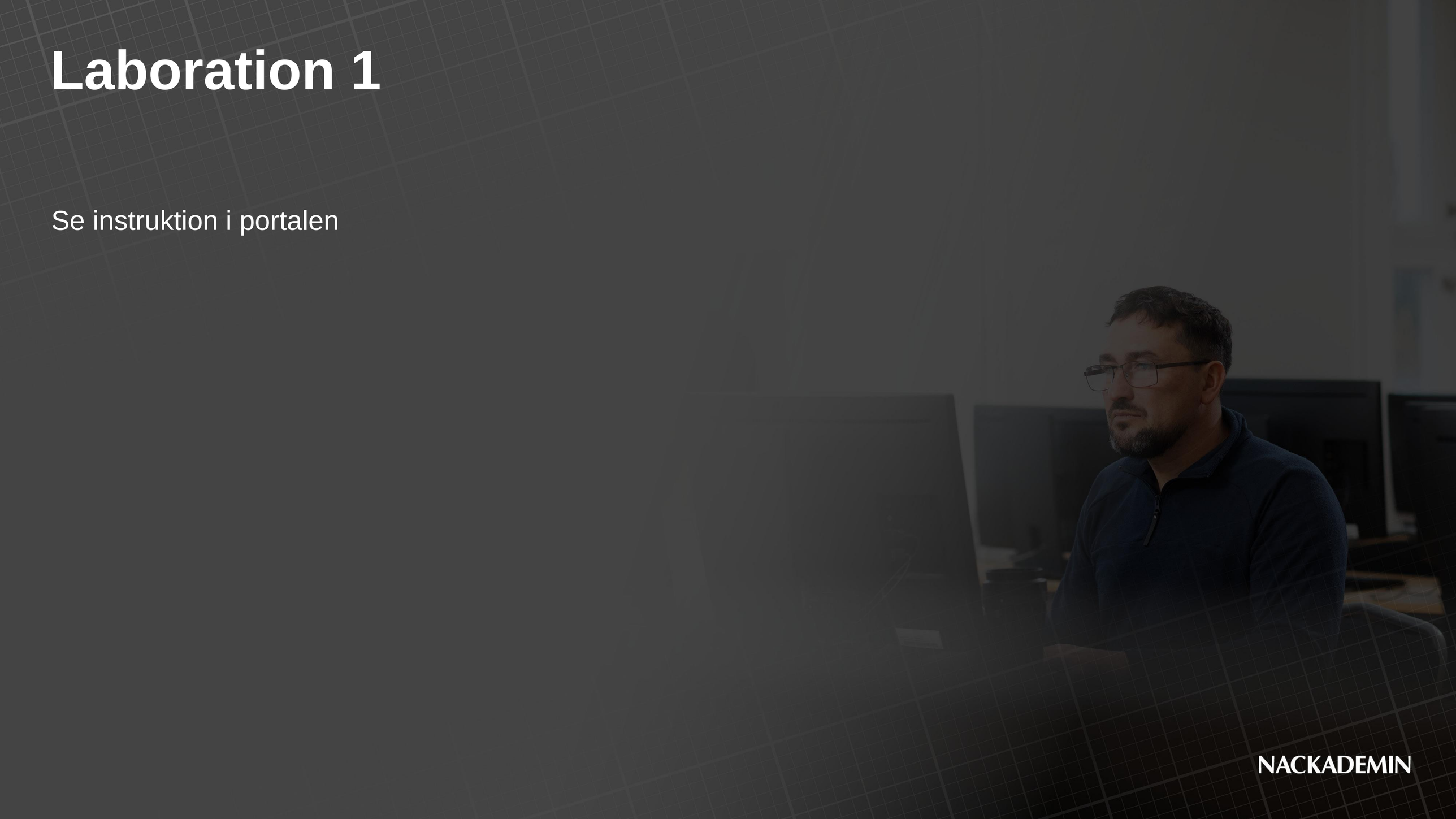
Det går också att slänga in **ssh user@server** innan **tar** nummer två.

Linux har också stöd för filsystem med inbyggd datakomprimering, zfs.

Det kan till och med vara snabbare än okomprimerad lagring!

Laboration 1

Se instruktion i portalen



NACKADEMIN

”Backup”

eller att effektivt kopiera skillnader i filträd

Vi såg att man kan kopiera filträd – med komprimering – med hjälp av tar.

rsync går steget längre genom att bara kopiera filer som är nya eller förändrade. **rsync** kan också radera filer på destinationen som tagits bort hos källan.

Eftersom rsync är så praktiskt för just att kopiera skillnader i filträd är det vanligt att använda för backup.

”Alla” har sitt eget rsync-skript som är det ultimata backup-skriptet.

***rsync** kan köra över ssh eller sitt eget protokoll. I det andra fallet måste servern man ansluter till köra **rsyncd**.*

rsync – några exempel

Från man-bladet:

```
Perhaps the best way to explain the syntax is with some examples:

       rsync -t *.c foo:src/

This would transfer all files matching the pattern *.c from the current directory to the directory src on
the machine foo. If any of the files already exist on the remote system then the rsync remote-update pro-
tocol is used to update the file by sending only the differences in the data.  Note that the expansion of
wildcards on the commandline (*.c) into a list of files is handled by the shell before it runs rsync  and
not by rsync itself (exactly the same as all other posix-style programs).

       rsync -avz foo:src/bar /data/tmp

This  would  recursively  transfer  all  files  from  the  directory  src/bar on the machine foo into the
/data/tmp/bar directory on the local machine. The files are transferred in "archive" mode, which  ensures
that  symbolic  links,  devices, attributes, permissions, ownerships, etc. are preserved in the transfer.
Additionally, compression will be used to reduce the size of data portions of the transfer.
```

rsync med hårda länkar

Nu kommer det äntligen – en praktisk användning av hårda länkar!

”rsync” har en intressant funktion där man kan köra den mot en tom destination, men låta den jämföra med en tidigare synkad mapp.

De filer som inte är ändrade från tidigare synk hårdlänkas då till destinationen.

? Varför vill man göra det?

Det ultimata rsync-skriptet™

```
#!/bin/bash

# Incremental backup based on rsync

DESTINATION=backups@backuphost:backups
NUMREVISIONS=5
LOGFILE=rsync.log

[ -f "revision.txt" ] || echo 0 > "revision.txt"
prevRevision=$(cat revision.txt)
revision=$(( ($prevRevision + 1) % $NUMREVISIONS ))
echo $revision > "revision.txt"

while read source; do
    rsync -va --delete --exclude-from=excludes.txt \
        --link-dest=../$prevRevision "$source" $DESTINATION/$revision/ >> $LOGFILE
done < <(egrep -v '^#|^\s*$' sources.txt)

rsync revision.txt $DESTINATION >> $LOGFILE
```

- Hur fungerar skriptet?
- Vilka problem kan det finnas med den här lösningen?
- Vad behövs för att kunna köra skriptet med ”cron”?

rclone

”rclone” är likt ”rsync”, med några viktiga skillnader: https://rclone.org

- Kan backa upp till många olika molntjänster
- Kan migrera mellan olika molntjänster
- Kan kryptera destinationen
- Kan montera arkiv i molntjänst som nätverksdisk

rclone till ”molntjänst”

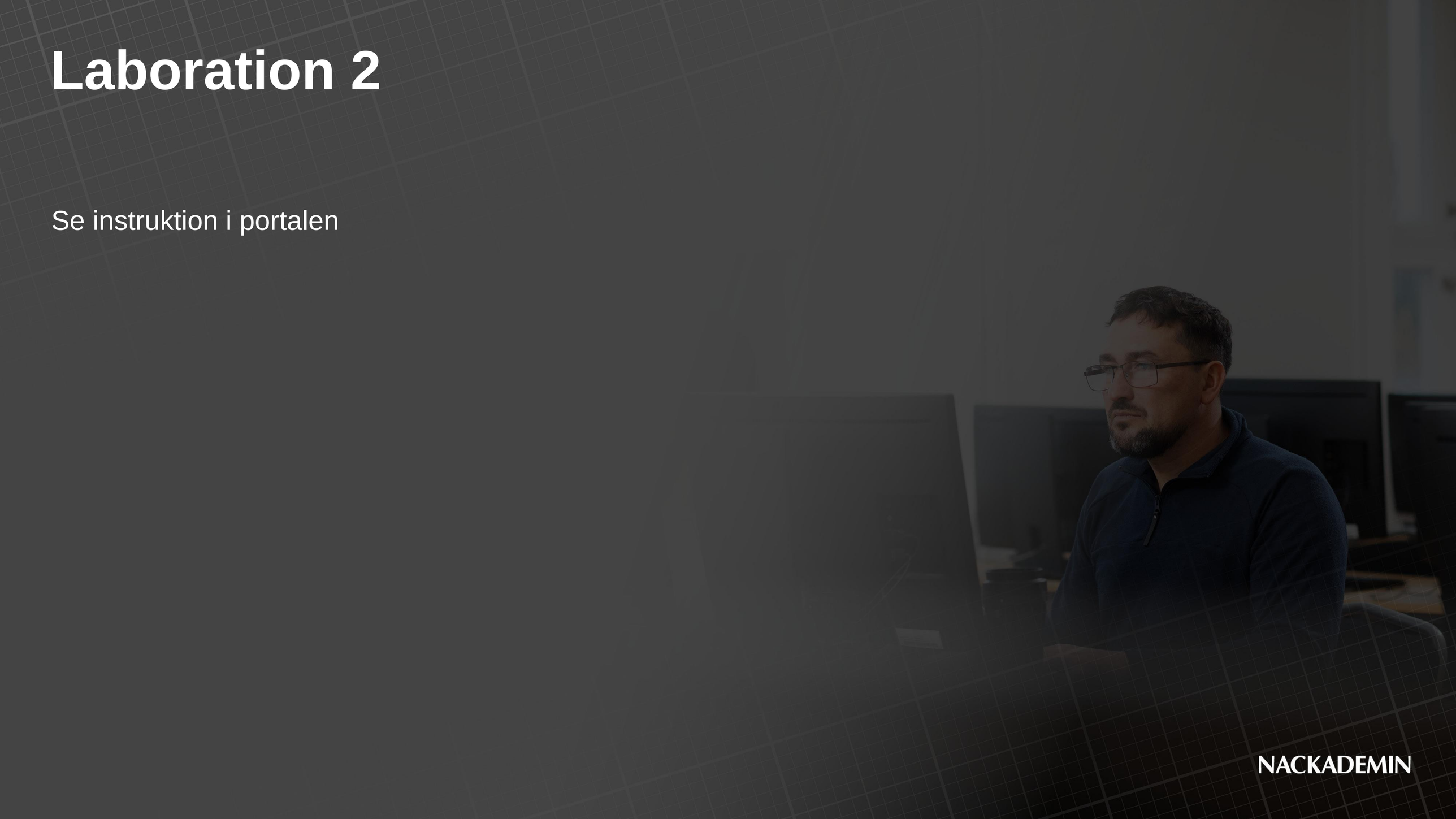
```
#!/bin/bash

# Incremental backup script based on rclone

DESTINATION=backupserver:backups2

while read source; do
    rclone sync -v --exclude-from=excludes.txt "$source" $DESTINATION &>> rclone.log
done < <(egrep -v '^#|^\s*$' sources.txt)
```

Destinationskatalogen (backups2) måste finnas, men kan skapas med:

```
$ rclone mkdir backupserver:backups2
```

Laboration 2

Se instruktion i portalen



NACKADEMIN

Summering

Idag har vi gått igenom vanliga metoder för filarkivering på Linux. Vi har lärt oss skilja på arkivering och komprimering.

Vi har även tittat på olika metoder för att utföra backup.

Nästa gång

Mål: Att veta hur loggar hanteras och hur man kan undersöka feltillstånd.

- Roterade loggar
- Diskutrymme
- Minne
- CPU
- Öppna filer
- Hängande processer

Stort tack!

NACKADEMIN