Nyttige verktøy: screen, ssh, scp

Typiske situasjoner

- Hvordan kan man samarbeide på kommandolinjen?
- Hva om jeg er koblet til via trådløst og mister signal mens jeg er midt inni noe?
- Ånei! Denne kommandoen tar kjempelang tid men jeg må gå nå...

Screen

- Et fabelaktig lite verktøy som gir deg et shell som overlever om du mister kontakten
- Du kan når som helst koble deg av og på dine screen sesjoner
- Alle kommandoer du kjører der vil fortsette å kjøre mens du er frakoblet
- Kan virke litt forvirrende i starten

Installere screen

- På et Debian basert Linux system installerer man screen slik:
  `apt-get install screen`
- Deretter har man en ny kommando man kan kjøre: `screen`

Bruke screen

Hint: ^ betyr control-tasten, [ ] betyr at noe er frivillig

- Opprette en ny screen sesjon:
  `screen -S navn`
  - Man havner så rett inn den nye sesjonen
- Gå ut av en screen sesjon
  `^a ^d`
- Liste opp kjørende sesjoner
  `screen -ls`
- Gå tilbake inn i en eksisterende sesjon:
  screen -r [-d] navn

Dele en screen sesjon

- En virkelig kraftig funksjon til screen er muligheten til å dele en sesjon mellom flere personer
  - De kan jobbe sammen i samme shell
  - De MÅ være logget inn som samme bruker
- Person 1: oppretter en ny screen sesjon:
  `screen -S navn`
- Person 2: hopper inn i samme sesjon:
  screen -x navn

Fjerndrift med ssh

.. og noen shell-triks

IMT3003 kyrre.Begnum@ntnu.no

Nøkkel-basert innlogging

- Hvis man slipper å taste passord ved ssh, kan man lettere bruke det i script
- Det er mulig å bruke asynkron kryptering til å få passord-løs innlogging
- Først må man generere nøkler for brukerenssh-keygen
- Deretter kopierer man den offentlige nøkkelen:
  `ssh-copy-id bruker@maskin`

Gi en kommando til SSH

- Skal du bare gjøre en enkel kommando eksternt, kan du gi den med som argument:
  `ssh maskin kommando`
- Ved flere kommandoer, kan man kjøre dem etter hverandre slik:
  `ssh maskin "kommando1;  kommando2"`

for-løkker i shell

- Dersom du skal kjøre samme kommando på flere maskiner, kan du lage en løkke:

```
for maskin in maskin1 maskin2 maskin3; do
ssh $maskin kommando; done
```

- Man kan selvfølgelig kjøre flere kommandoer inni løkken
- Skal man heller telle, blir kommandoen slik:

```
for tall in $(seq 1 6); kommando $tall; done
```

Aliaser

- Lei av å skrive lange kommandoer? Lag deg aliaser
- Åpne filen .bashrc i brukerens hjemmekatalog og legg til, f.eks:

```
alias wwwroot="ssh root@www"
alias checkall="for ..... ; done"
```

- Endringene gjelder fra *neste* shell
- Sjekk hvilke aliaser som finnes med kommandoen:

```
alias
```

grep

- Kommandoen grep kan brukes til å sortere ut linjer som inneholder en bestemt tekst
- Det er svært vanlig å bruke grep sammen med andre kommandoer:

```
cat minfil | grep mintekst
kommando | grep mintekst
```

Repetere kommandoer

- Man kan kjøre samme kommando omigjen og omigjen ved hjelp av kommandoen watch:

```
watch -n 3 "kommando"
```

- Kommandoen blir gjentatt helt til man trykker ^c
- Dette egner seg kun til "lette" kommandoer
  - Ved tyngre kommandoer kan man bruke lengre intervall

Sikker kopiering med scp

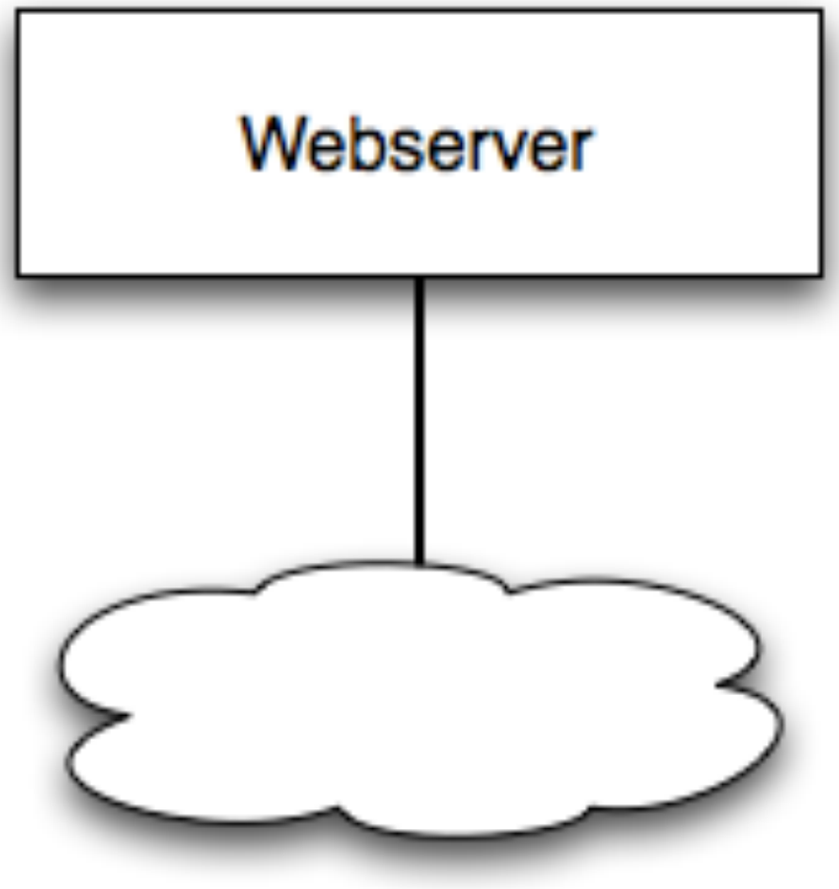
- scp er en kommando som bruker ssh til å kopiere filer:
  - Pushe en fil:
    `scp minfil  bruker@maskin:[mappe]`
  - Hente en fil:
    `scp bruker@maskin:[mappe/]minfil  .`
- Fordelen er at man ikke trenger å sette opp en egen tjeneste
- scp klient i windows finnes også: WinSCP

Tjenestearkitekturer

Informasjon Vs. logikk

- Et system hos en organisasjon kan grovt deles inn i *informasjonen* (data) og *logikken* (selve programmet)
  - 'Logikken' bestemmer hva som skal skje med informasjonen og blir programmert av systemutviklere
  - 'Informasjonen' må håndteres etter bedriftens egen politikk og blir passet på av systemadministratoren

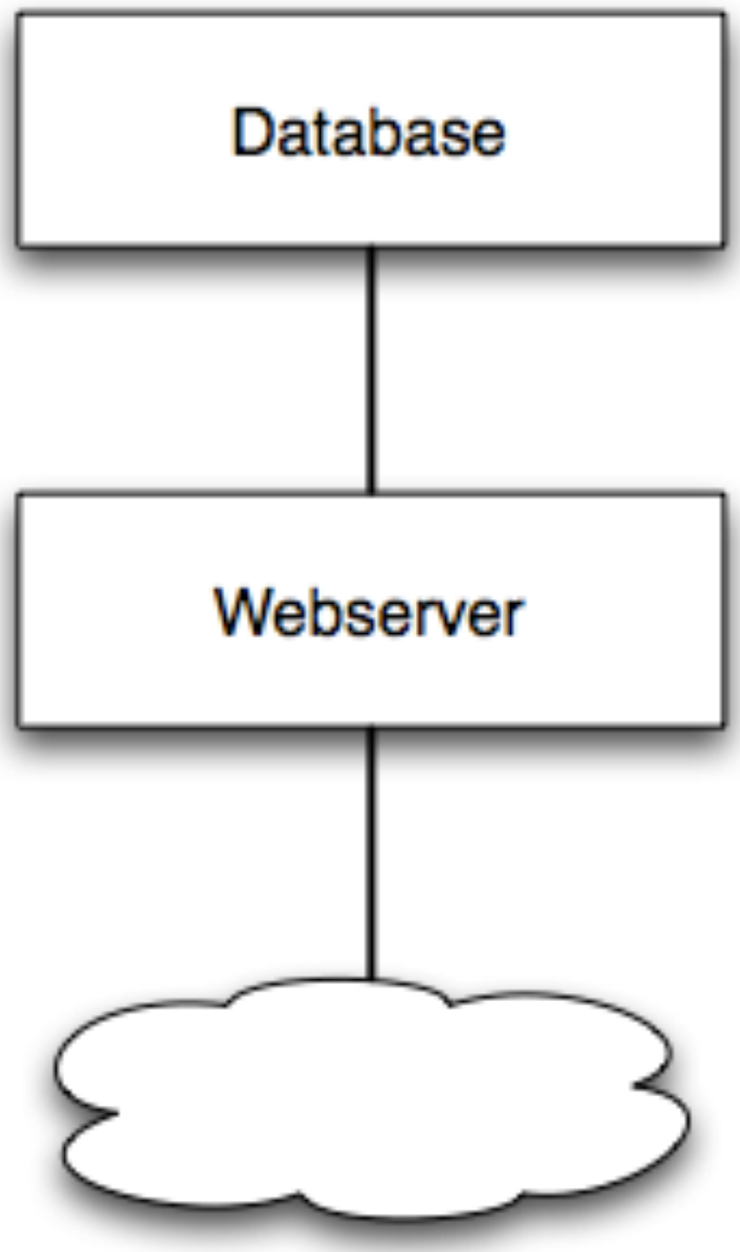
I begynnelsen...



Kun en applikasjon

- Informasjon og logikk er knyttet sammen
- Filsystemet brukes til lagring
- Applikasjonen må selv håndtere transaksjoner og serialisering
- Søk må implementeres i applikasjonen

To-lags arkitektur



Databasens funksjon

- Transaksjoner og serialisering av operasjoner
- Tilgangskontroll
- Backup og redundans
- Effektivitet i søk (Indeksering og søketrær)
- Hurtiglagring (caching) av hyppig brukt infromasjon
- Komplisert datauthenting (SQL)

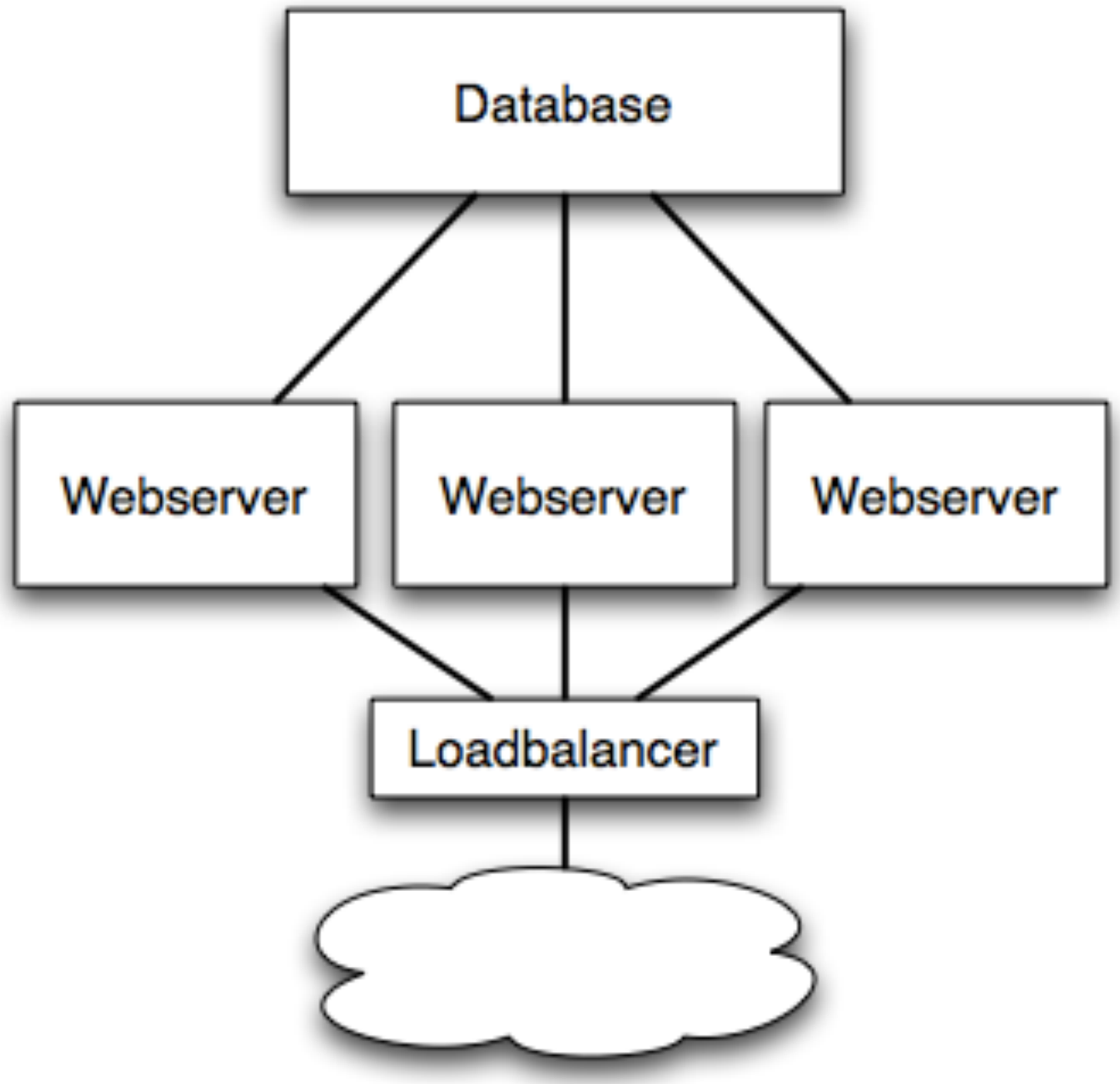
To-lags arkitektur

- Informasjon og logikk er adskilt
- Dataintegritet ivaretas på databasenivå
- Tilgangskontroll mot brukere ivaretas på applikasjonsnivå

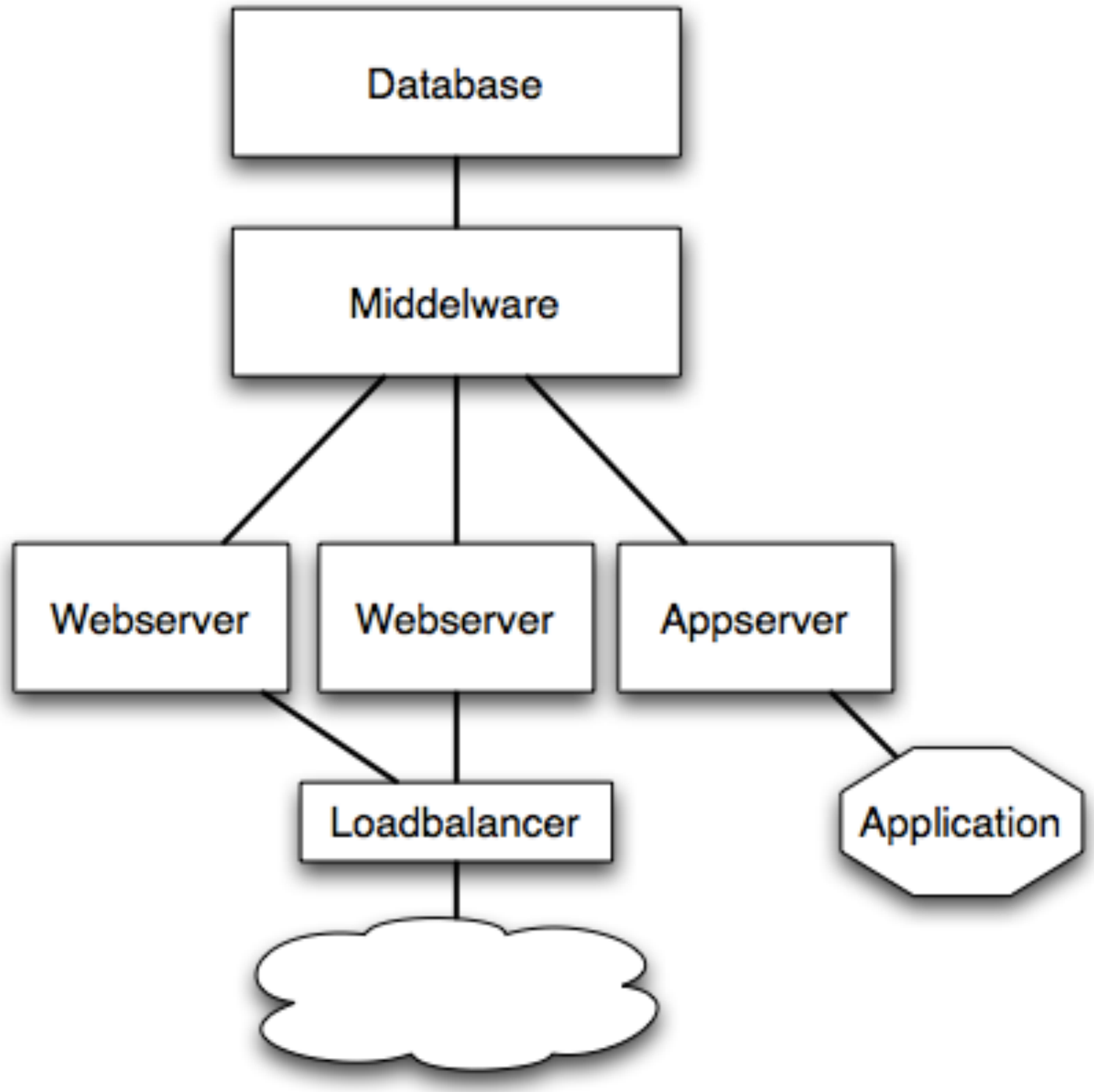
To-lags arkitektur

- Applikasjonslogikk kan bygges inn i databasen via datamodellen
  - Applikasjonen er database-bevisst
  - Databasen kan brukes til å "lese" data manuelt
  - Lettere å bygge ut arkitekturen grunnet serialisering og transaksjoner

Skalert to-lags arkitektur



Tre-lags arkitektur



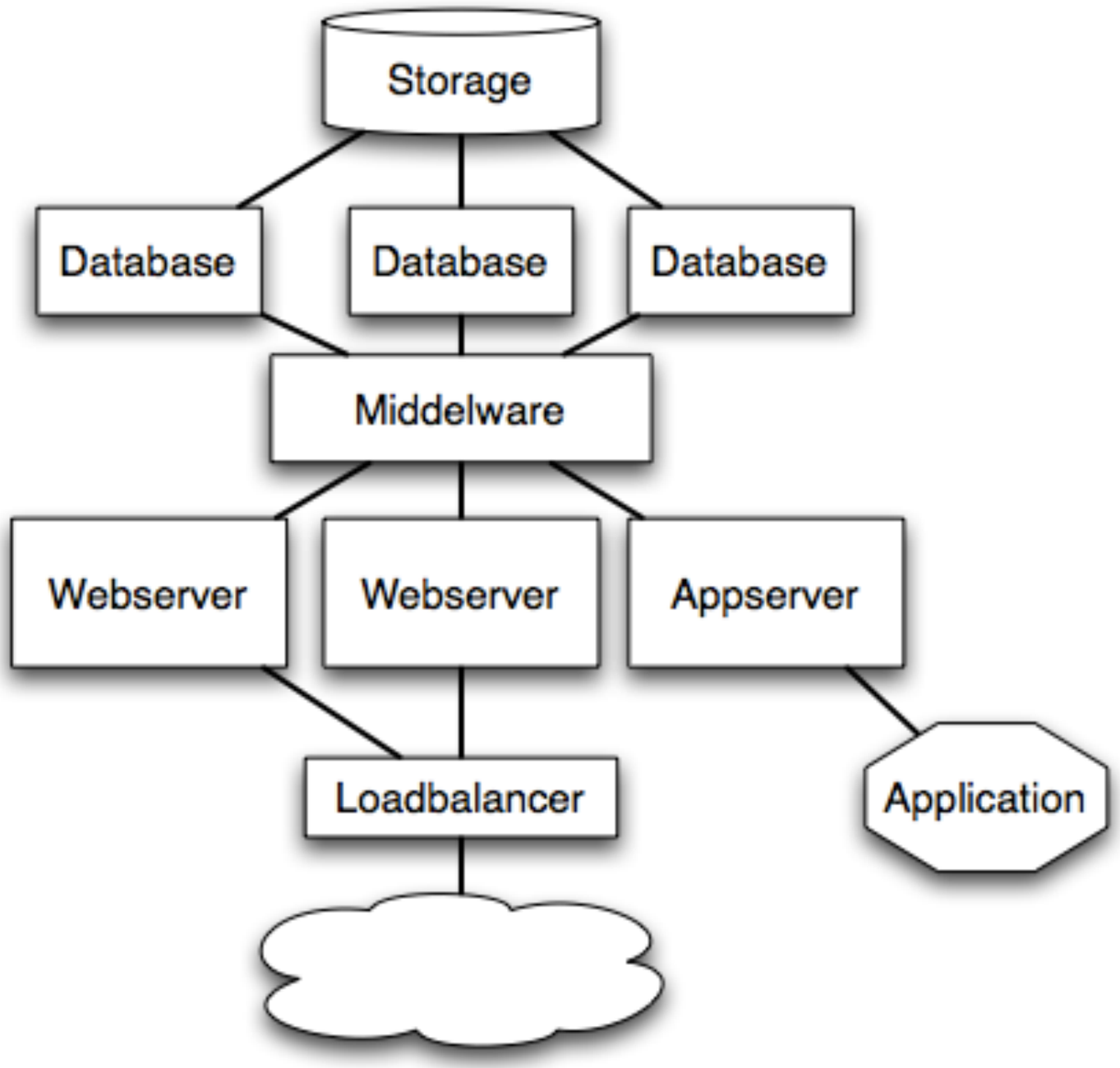
Tre-lags arkitektur

- Middelvare deler ut objekter eller avanserte datastrukturer istedenfor data tupler
- Applikasjonslogikk er sannsynligvis integrert i objektene (f.eks via objekt funksjoner)
- Applikasjonslaget er database-ubevisst

Tre-lags arkitektur

- Databasen ivaretar middelvarelogikk og ikke direkte applikasjonslogikk
- Informasjon er ikke nødvendigvis “lesbar” via en databaseklient
- Backup ivaretas av databasen
- Det er mulig å konsolidere flere gamle databaser "bak" en middelvare for å fremstå som én platform

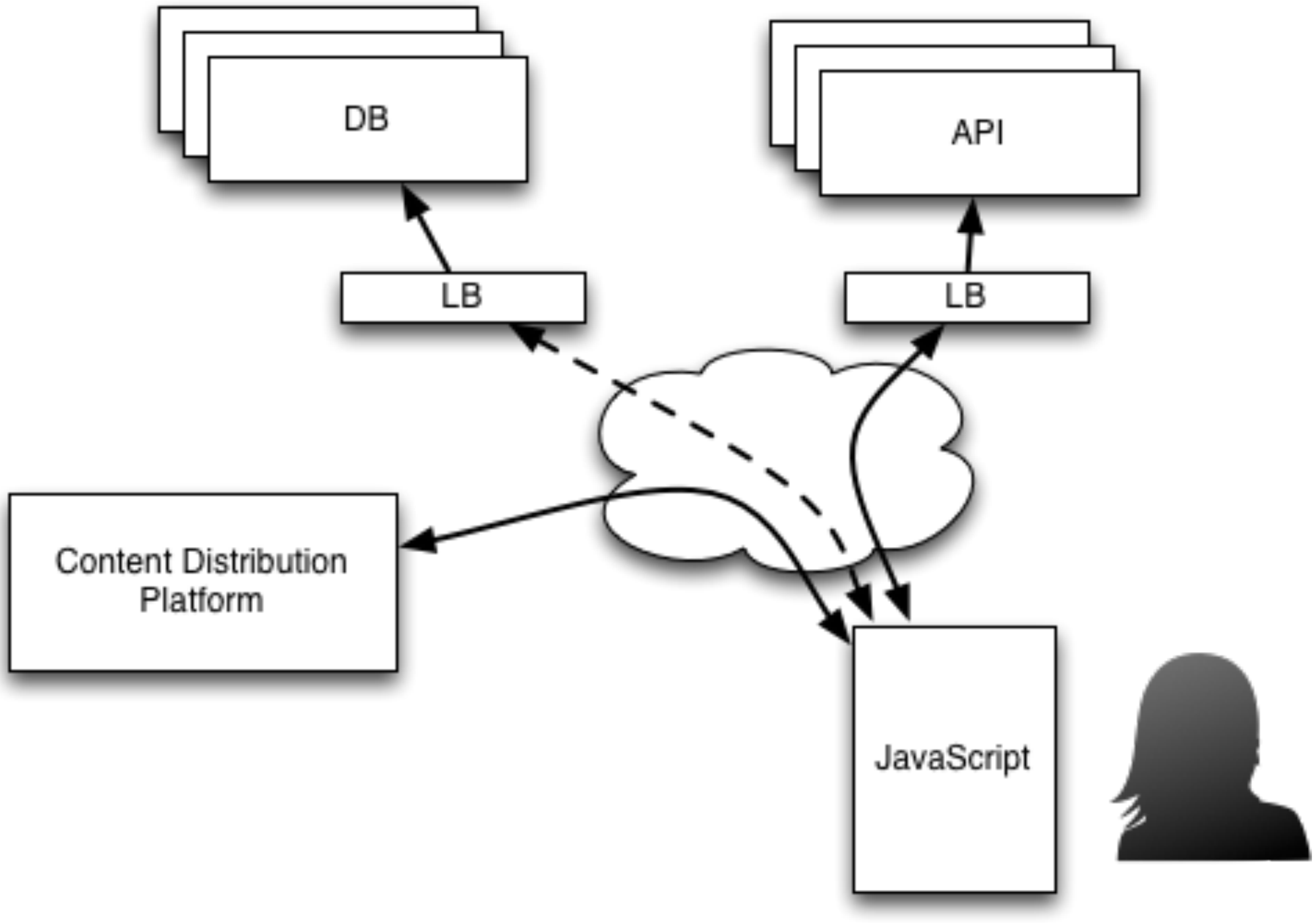
Tre-lags arkitektur med SAN



Tre-lags arkitektur med SAN

- Backup kan nå også skje via filsystemet (SAN)
- SAN'et er database ubevisst
- Noen databaser har egen SAN-/ lagringsteknologi

Moderne web-app arkitektur



Moderne web-app arkitektur

- Det meste av presentasjon og logikk er flyttet til klienten selv (browseren)
- Noe logikk er tilgjengelig via en API (REST, etc)
  - Dette skaper et ekstra økosystem med tredjeparts applikasjoner
- I noen tilfeller er applikasjonslogikken tilbake i databasen igjen (f.eks couchDB)
- Vanlig arkitektur i nye startups (eks. Trello.com) men også i "eldre" web-apps (Google Docs)

Observasjoner

- Jo mer distribuert arkitekuren er, desto mer kan man distribuere driften av den
  - Applikasjonsdrift
  - Databasedrift
  - SAN/Infrastrukturdrift
- Dette krever økt koordinering mellom gruppene (noen ganger er det forskjellige firmaer)
- Flere kokker...

Web-baserte applikasjoner

Hvordan fungerer "webben"

- En trend i såkalte web 2.0 applikasjoner, er at alt går over HTTP
- Applikasjonen eller websiden er bygget opp av mange elementer
  - Bilder / animasjoner
  - Reklame
  - Navigasjon
  - Tekst
- Å se en webside betyr at man må laste ned mer enn en "fil"
- Det er ikke sikkert at alle disse filene kommer fra samme sted / server

HTTP

- Når en nettleser skal laste ned en side, skjer følgende:
  - HTTP GET forespørsel til "hoved"-siden
  - HTML'en gjennomgås og påfølgende HTML GET forespørsler blir gitt til alt andre materialet
    - stilark (css)
    - bilder
    - javascript
    - nye sider (med enda flere elementer)

Linker og atter linker

- Gir man ressursene slik som bilder direkte eller url-er til bilder?
  - Hva er fordelen / ulempen med dette?
- Muligens en mengde database oppslag under panseret for hver side som skal vises

Eksempel: Hvor mange linker har https://www.ntnu.no's hovedside?

- Hvordan kan man finne det ut?

Klient-side vs server-side

- Funskjonalitet og interaktivitet kan leveres på to måter:
  - Klient-side: Javascript / AJAX
  - Server-side: PHP / .NET / Python
- Vi skal bevege oss mest på server-siden (back-end)

Lastbalansering

Skalering

- Vi møter ofte situasjoner hvor ytelsen på systemet vårt ikke er tilstrekkelig
  - Nytt bruksmønster
  - Flere brukere
  - Mer data
- Siden vi sjeldent kan endre programvaren, løser vi problemet ved å skalere
  - Øke ytelse til maskinpark / infrastruktur

Vertikal skalering

- Vi øker kraften til eksisterende maskinvaren:
  - Øker minne
  - Raskere / flere prosessorer
  - Raskere disk
  - Bedre linje
- En "enkel" løsning dersom det er mulig
  - Krever nedetid
- Krever ingen endring i programvaren (så lenge programvaren kan dra nytte av det)

Horisontal skalering

- Flere servere av samme type
- Lastbalansering mellom dem
- Lettere å gå fra 2 til 4 enn fra 1 til 2 servere
  - Krever at programvaren takler det
  - Krever infrastruktur
  - Hvilket lag skal man skalere?
    - Må finne flaskehalsen først

Lastbalansering

- Vi deler belastningen mellom flere servere
  - Beslutningen tas av en lastbalanserer eller lignende
  - Forutsetter at alle klienter potensielt kan få utført jobben av en vilkårlig server
    - Typisk for web, epost
    - Ikke godt egnet for spillservere
- Mange balanseringsalgoritmer å velge mellom

Regional lastbalansering

- Man blir videresendt til den serveren / serverparken som er geografisk nærmest klienten
- Håndteres enten av DNS eller lastbalanserer
- Egnet for større infrastrukturer
  - Men ikke umulig for mindre infrastrukturer

Lokal lastbalansering

- Klassisk oppsett med lastbalanserer og servere i samme serverpark
- Lastbalanserer kan være dedikert hardware eller en standard server
- Enkleste løsningen
  - Gir mindre robusthet enn regional lastbalansering

Innholdsbasert lastbalansering

- Istedenfor at alle serverne gjør det samme, fordeler man innholdet til noen spesialister
  - Noen servere til statisk innhold
    - Bilder / CSS / Javascript filer
  - Noen servere til dynamisk innhold
    - PHP / ASP .NET
  - Eksempel:Varnish
- Kan kombineres med regional og lokal lastbalansering
  - Eks: Amazon Cloudfront

Utfordring: Tilstandskontroll

- En typisk utfordring er i hvor stor grad man må håndtere tilstanden / tilkoblingen til klienten
  - Nettbanker har lange sesjoner
  - Lolcats.com har ikke samme behov
- Lastbalansereren må støtte sesjoner
  - Enkel løsning er å alltid sende samme IP til samme server

Apache installasjon

Apache

- Apache er en populær webserver
- Finnes til mange platformer
- Lett å sette opp hvis man kun skal ha én site
- Vi skal lære mer om Apache senere, men trenger det til neste gang

Installasjon med apt-get

- Selve webserveren

```
apt-get update
apt-get install apache2
```

- PHP og MySQL støtte

```
apt-get install libapache2-mod-php php-mysql
apt-get install mysql-client
```

- Sjekk at den kjører fra manager:
  wget -q -O - http://<maskin_ip>/
  (Det skal være 'stor o', ikke null i linjen over)
- Skal gi: "It works!" med en del HTML rundt

Test om PHP fungerer

- Lag en fil i /var/www/html som heter index.php
- Filen skal inneholde følgende:

```
<?php
phpinfo();
 ?>
```

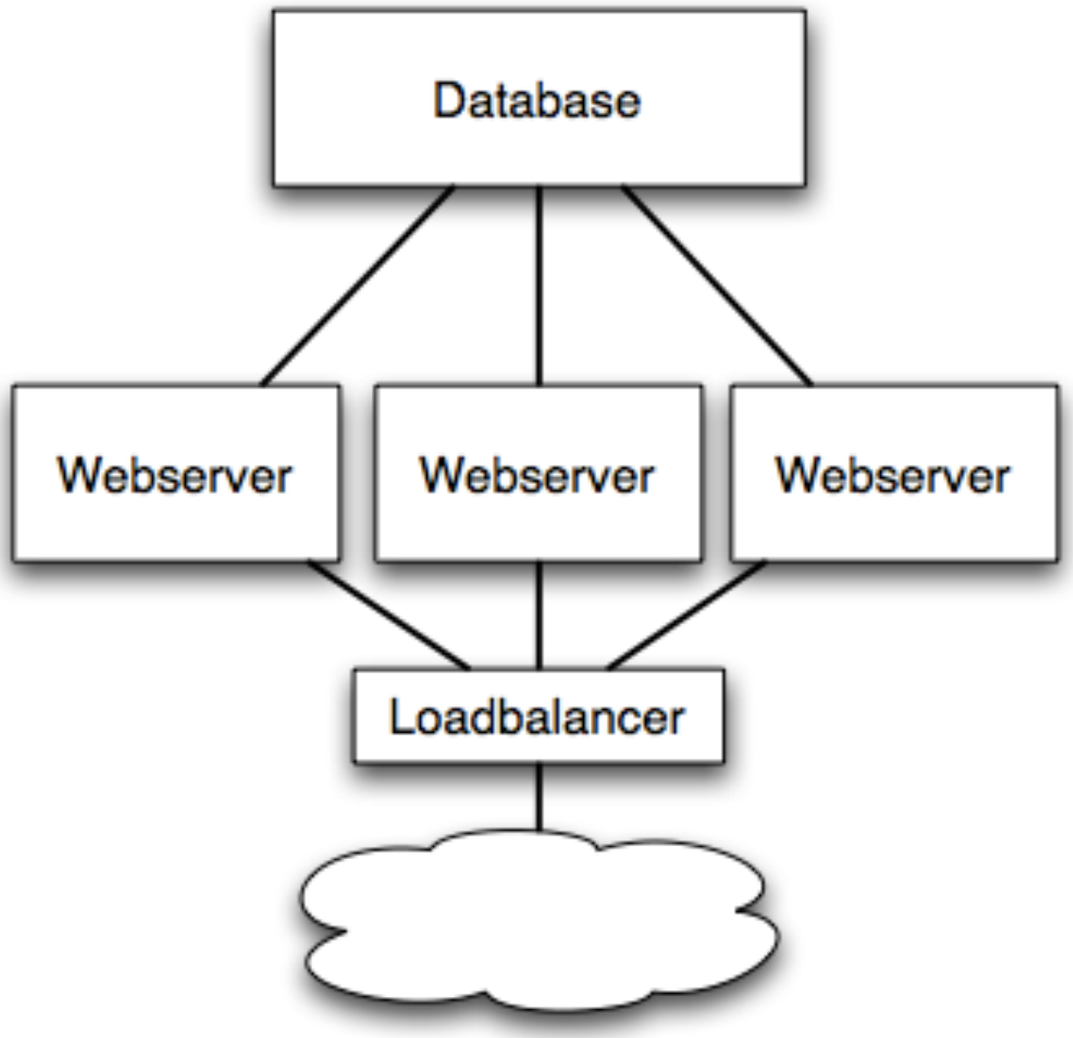
- Restart apache:

```
service apache2 restart
```

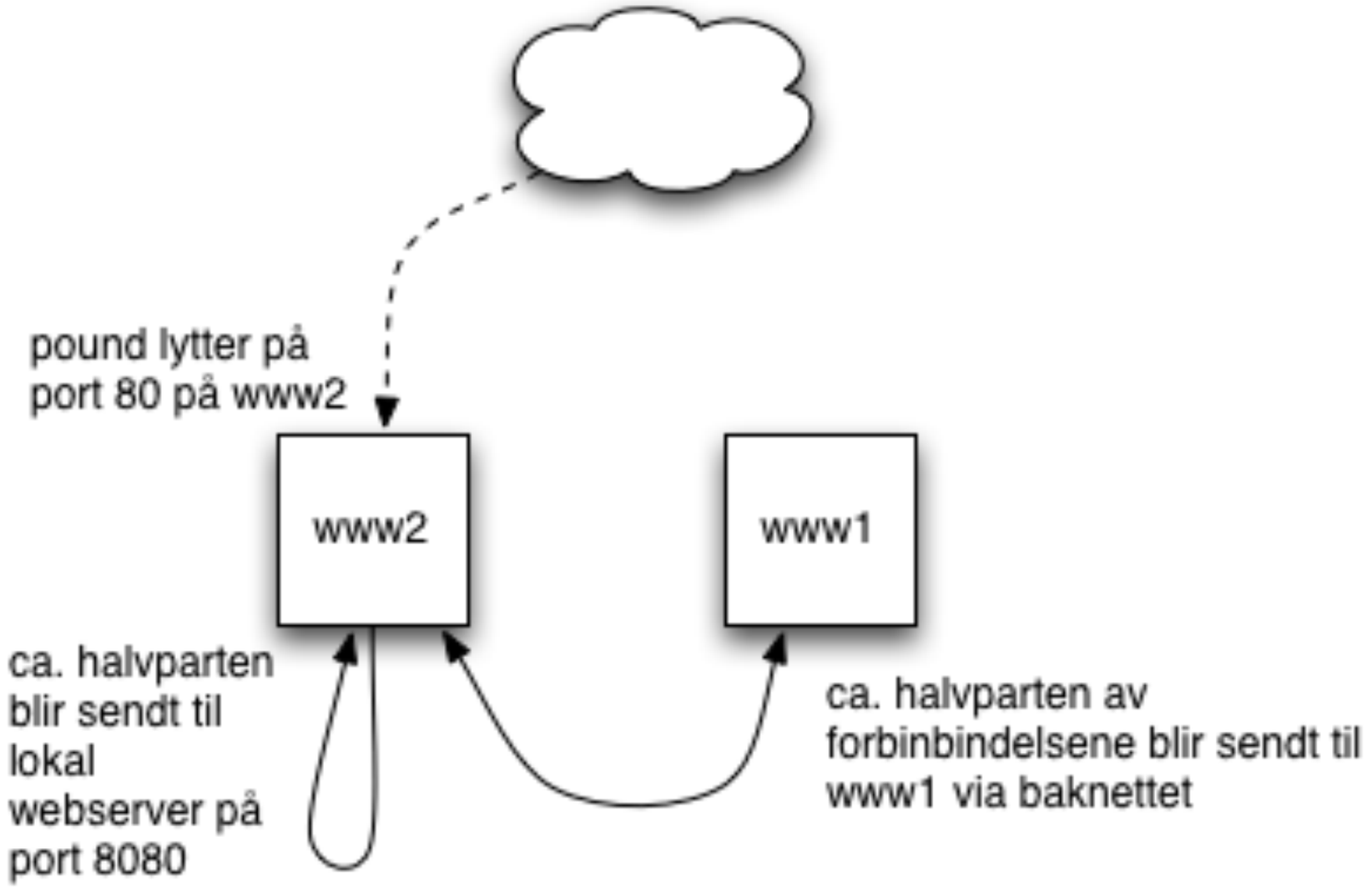
- Test: `wget -q -O - http://<maskin_ip>/index.php`
- Resultatet skal være en infoside.

Lastbalansering på web

Klassisk lastbalanserer



Alternativ lastbalansering med to maskiner



Pound

- Lastbalanserings motor for Linux, utviklet av Apsis Security: http://www.apsis.ch/pound/
- Støtter mange forskjellige reverse-proxy muligheter
  - HTTP / HTTPS rproxy eller fail-over
  - Balansering basert på URL, ID, klient osv.
- Stiller ingen krav til webserverne
- Ikke det eneste alternativet, men enkelt å sette opp

haproxy

- Verdenskjent lastbalanserer med høy ytelse
- Fortsatt videreutviklet - haproxy.org
- Fungerer på de fleste UNIX/Linux OS
- Har installasjonspakker på "alle" Linux distroer

haproxy installasjon

- Først, legg til offisielt repository:

```
sudo add-apt-repository -y ppa:vbernat/haproxy-1.8
```

- Oppdater pakkelisten og installer

```
apt-get update
apt-get install haproxy socat
```

Konfigurere haproxy

- Konfigurasjonsfilen er `/etc/haproxy/haproxy.cfg`
- Denne er nokså komplisert, men det fungerer å kun legge til tre viktige deler:
  - En frontend definisjon
  - En backend gruppe
  - Et administrasjonspunkt

Lage en frontend

- I bunnen av haproxy.cfg, legg til følgende

```
frontend bookface
    bind *:80
    mode http
    default_backend nodes
```

Definere backend noder

- Legg til i bunnen av haproxy.cfg

```
backend nodes
    mode http
    balance roundrobin
    server web01 web01-ip:80 check
    server web02 web02-ip:80 check
```

Lage adminpunkt

- haproxy har mulighet for fjernstyring og status, men det må også legges til
- I haproxy.cfg, legg til nederst:

```
listen stats
    bind *:1936
    stats enable
    stats uri /
    stats hide-version
    stats auth someuser:password
```

- Nå er det mulig å åpne

  `http://<lastbalanserer-ip>:1936`

  for å se status

Admin muligheter fra kommandolinjen

- Det finnes også en lokal fil-socket som kan brukes
  - Det er lettere å bruke til script etc.
- Kjør følgende som root:

```
echo "show stat" | socat unix-connect:/var/run/haproxy/admin.sock stdio
```

- Andre kommandoer
  - show info, show errors, show sess

Test lastbalansering

Lag en enkel PHP fil: test.php

```
<?php
echo "your IP: " . $_SERVER['REMOTE_ADDR'] . " ";
echo "served by: \"" . $_SERVER["SERVER_ADDR"] . "\"\n";
?>
```

- Legg den på begge serverne i /var/www
- Kjør kommando:

  `wget -O - -q http://utside-ip/test.php`
- Man kan også kjøre det i en løkke:

  `while true; do wget -O - -q http://utside-ip/test.php; sleep 2; done`

Nye utfordringer

- Ved lastbalansering mister webserveren informasjon om hvem klienten er
  - Dette informasjonen er viktig mht. cookies, tilgang, sessions osv.
  - Man kan få tilbake den informasjonen ved å isntallere tillegsmodulet mod_rpaf i apache2