Reititys

- Oletusreitin asetus (yleensä automaattinen mutta...):

  ip route add default via reititin

- Jos johonkin koneeseen tai verkkoalueeseen ei pääse oletusreitin kautta, sille pitää määritellä oma gateway. Esim.

  ip route add 192.168.122.0/24 via 172.20.208.17

  ohjaa aliverkkoon 192.168.122.0/24 lähtevät paketit lonka7 -koneen kautta.

  lp route add 192.168.122.19 via 172.20.208.17

  vastaavasti mutta vain yksittäiselle koneelle.

- Käytössä olevaa reititystaulua voi katsella:

  ip route [show]

- Reitityksen poisto (argumentteina samat kuin ip route add -komennossa annettiin tai mitä ip route show näyttää):

  ip route delete ...

Reititys 2

- Reittejä voi laittaa /etc/network/interfaces -tiedostoon tähän tapaan:

  up ip route add 192.168.2.0/16 via router1

  ”up” määrää, että komento suoritetaan kun ko. interface käynnistetään (ifup)
- Jotta gateway-kone välittäisi paketteja eteenpäin se pitää erikseen sallia siellä:

  echo 1 > /proc/sys/net/ipv4/ip_forward # tai

  sysctl -w net.ipv4.conf.all.forwarding=1
  - Tarkistus: cat /proc/sys/net/ipv4/ip_forward # tai

    sysctl net.ipv4.conf.all.forwarding
  - Asetus pysyväksi: net.ipv4.ip_forward=1 tiedostossa /etc/sysctl.conf
  - Gateway-koneen mahdollisessa (softa)palomuurissa tarvitaan myös säännöt tätä varten
- Ellei gateway-kone ole omassa aliverkossa, sinne pitää määritellä reitti ensin!
- Kahden samaa privaattiosoitealuetta käyttävän aliverkon yhdistäminen reitittämällä ei onnistu; jos mahdollista kannattaa käyttää eri osoitealueita, muuten tarvitaan monimutkaisempia palomuurisäätöjä (NAT) tai tunnelointia tms.

Reititysesimerkki

- JY:n julkinen verkko: 130.234.0.0/16, labraverkon alue 130.234.208.0/23
- Kurssin sisäverkko: 172.20.0.0/16

  Sisäverkon rajapalomuuri: 172.20.0.1, 130.234.254.76
  - Palomuurin julkinen verkkoalue: 130.234.208.0/23
  - Palomuurin sisäinen verkkoalue: 172.20.0.0/16 (ja pari muuta)
  - Sisäverkon dns/dhcp-palvelin (Raspberry Pi): 172.20.0.2

  lonka6: 130.234.208.16, 172.20.208.16, 192.168.123.1
  - Lonka6:n sisäinen (NAT-)verkko: 192.168.123.0/24

  lonka7: 130.234.208.17, 172.20.208.17, 192.168.122.1
  - lonka7:n sisäinen (NAT-)verkko: 192.168.122.0/24

  pouta: 130.234.208.10, 172.20.208.10

  tt1: 130.234.209.19, 172.20.209.19 (siltaverkko)

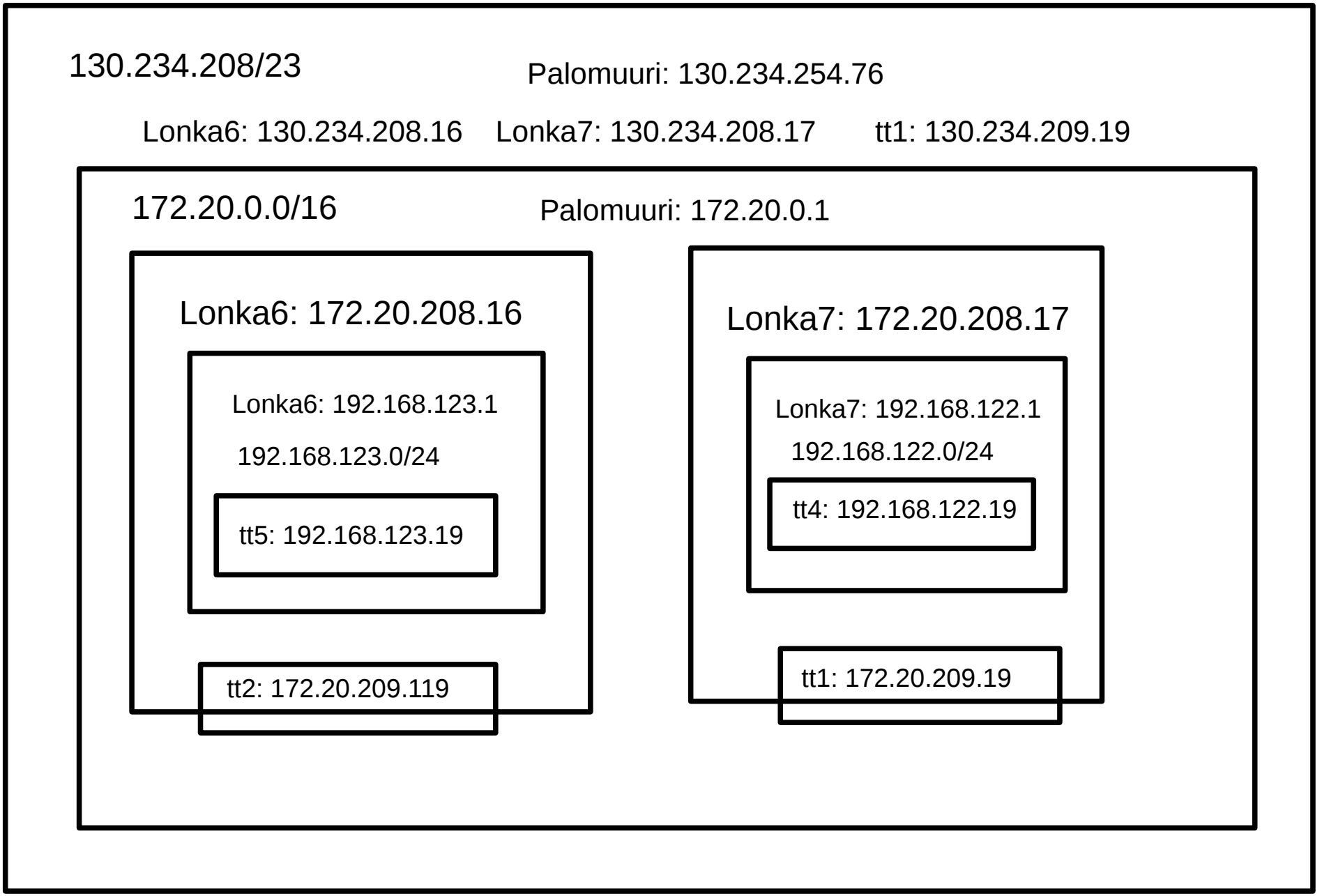
  tt2: 130.234.209.119, 172.20.209.19 (siltaverkko)

  tt3: 172.20.209.219 (siltaverkko, ei julkista IP:tä)

  tt-bak: 172.20.210.19 (siltaverkko, ei julkista IP:tä)

  tt4: 192.168.122.19 (NAT-verkko, ei julkista IP:tä eikä ulomman sisäverkon IP:tä)

  tt5: 192.168.123.19 (NAT-verkko, ei julkista IP:tä eikä ulomman sisäverkon IP:tä)


130.234.208/23
Palomuuri: 130.234.254.76
Lonka6: 130.234.208.16
Lonka7: 130.234.208.17
tt1: 130.234.209.19
172.20.0.0/16
Palomuuri: 172.20.0.1
Lonka6: 172.20.208.16
Lonka6: 192.168.123.1
192.168.123.0/24
tt5: 192.168.123.19
tt2: 172.20.209.119
Lonka7: 172.20.208.17
Lonka7: 192.168.122.1
192.168.122.0/24
tt4: 192.168.122.19
tt1: 172.20.209.19

Reititysesimerkki...

- Samassa aliverkossa olevat koneet näkevät toisensa suoraan
- Palomuuri reitittää omien aliverkkojensa ja julkiverkon välillä
- VM-alustakoneet (lonka*, pouta) reitittävät aliverkon 172.20 sisällä
- VM-alustakoneet reitittävät myös omien sisäverkkojensa (192.168.x) ja (omasta näkökulmastaan ulkoisen) aliverkon 172.20 välillä
- Bridge-verkkoa käytettäessä virtuaalikoneen oletusreittinä voi käyttää joko palomuuria (172.20.0.1) tai jotakin alustakoneista (tyypillisesti alustakonetta jossa se itse on). Jos VM on NATattuna alustakoneen sisäisessä (192.168.x) -verkossa, oletusreitin pitää olla alustakoneen sisäinen osoite (192.168.x.1)
- Alustakoneiden sisäverkkoihin pääsee vain ko. alustakoneen kautta, siis käyttämällä sitä joko oletusreittinä tai erikseen määriteltynä reittinä kyseiselle sisäverkolle (kumpikin edellyttää, että alustakoneen palomuuri sallii tämän, ulkoverkosta tultaessa myös rajapalomuuri(t))

Reititysesimerkki...

- Koneet tt1, tt2, tt3 ja tt-bak pääsevät toisiinsa suoraan, muualle käyttäen (oletus)reitittimenä joko palomuuria tai alustakoneitaan, koska niissä on käytössä siltaverkko (bridged net)
- Koneissa tt4 ja tt5 on käytössä NAT-verkko, josta ne näkevät suoraan vain oman alustakoneensa sisäisen (NATatun) verkon (192.168.x.0), oletusreittinä toimii vain aina oman alustakoneestaan sisäinen IP (192.168.x.1)
- Alustakoneet tarjoavat sisäverkkoonsa myös nimipalvelun, joten tt4:ssä nimipalvelinkin on 192.168.122.1; se voisi käyttää ulkoistakin nimipalvelua (esim. 172.20.0.2), mutta silloin se ei saisi omaa nimeään (eikä muita NAT-verkkonsa koneita) nimipalvelusta
- Jotta NAT-sisäverkon ulkopuoliset koneet pääsisivät sen sisälle, niissä pitää määritellä reitittimeksi sinne ko. alustakone (tt4:n tapauksessa siis lonka7, ja nimenomaan sen ”ulompi sisäinen” IP 172.20.208.17)
- Alustakoneen käyttäminen oletusreittinä toimii myös siltaverkon (tt1, tt2 &c) kanssa – kunhan ko. kone pysyy päällä (ongelma migraation kanssa)

Reititysesimerkki...

- Kone tt4 näkee oletusasetuksilla kaiken paitsi lonka6:n sisäverkkoa 192.168.123.0 (ja tt5:ttä), vastaavasti tt5 näkee kaiken paitsi lonka7:n sisäverkkoa (ja tt4:ää)
- Jotta kone tt1 näkisi myös tt4:n, kaksi vaihtoehtoa:
  - Oletusreitiksi lonka7 (172.20.208.17)
    - /etc/network/interfaces:
      gateway 172.20.208.17
  - Erillinen reititys lonka7:n kautta sen sisäverkkoon (tai haluttaessa vain tt4:ään)
    gateway 172.20.0.1
    up ip route add 192.168.122.0/16 via 172.20.208.17
- Jotta kone tt4 näkisi tt5:n:
  - Oletusreitin oltava kuitenkin aina lonka7 (192.168.122.1)
  - Aliverkolle 192.168.123.0/16 määriteltävä erikseen reitti, reitittimenä lonka6 (172.20.208.16)
    gateway 192.168.122.1
    up ip route add 192.168.123.0/16 via 172.20.208.16

ip

- Uusi yleiskomento verkkoasetusten säätämiseen, tarkoitettu korvaamaan mm. komennot ifconfig, route, netstat ja arp (mutta Linux-spesifi, jälkimmäisiä tarvitaan yhä mm. *BSD:ssä, ja ne toimivat useimmissa Linux-jakeluissakin yhä)
- Paljon toimintoja, yleinen syntaksi

  ip [*optiot*] *kohde* { *komento* }

  missä kohde voi olla jokin seuraavista:

  link, address, addrlabel, route, rule, neighbour, ntable, tunnel, tuntap, maddr, mroute, mrule, monitor, xfrm, netns, l2tp, tcp_metrics

  (useimmat voi lyhentää, esim. ”ip ro” jne)
- Yleisiä optioita -s[tatistics]: näytä tilastoja, -r[esolve]: hae nimet DNS:stä (vanhoissa route, netstat jne komennoissa dns-haku tehtiin oletuksena ellei -n -optiota annettu, ip:ssä käytetään numeroita ellei -r:ää ole annettu)

ip route

- Reititystaulun tutkimiseen ja muokkaamiseen (korvaa vanhan route-komennon). Yleisimmät käyttötavat ja joitakin esimerkkejä:

ip route [show]

- Näyttää reititystaulun; lyhennettävissä esim. ”ip ro”

ip route add *address[/mask]* via *router* [dev *device*]

ip route add 192.168.122.0/24 via 172.20.208.17 dev eth0

ip route delete ...

- Argumentit kuten add-komennossa (tai kuten show näyttää)

ip route change ...

ip route change default via 172.20.208.17

ip route replace ...

- Kuten change mutta ei valita vaikka reittiä ei ennestään olisi

ip route get *address*

- Selvittää reitin yksittäiseen osoitteeseen

ip route save >*file*

- Kuten show mutta tulostus binäärimömmö, jota restore ymmärtää

ip route restore <*file*

ip link, ip address

- Verkkolaitteiden ja osoitteiden asetukset (korvaavat ifconfig-komennon). Paljon alakomentoja ja optioita, yleisimpiä:

  ip link [show]

  ip -statistics link show # lyhyemmin ip -s li

  ip link set dev eth0 up

  ip li set dev eth0 arp off

  ip address [show]

  - ”show” on oletus ja voidaan jättää pois ellei lisäargumentteja ole, kuten näissä:

  ip addr show eth0

  ip add show up

  ip ad add 192.168.5.6/24 dev eth0 # ip alias = toinen ip samalle interfacelle

  ip ad broadcast 192.168.5.255 dev eth0

  ip ad delete 192.168.5.6/24 dev eth0

ip neighbour

- ”Naapuritaulun” käsittely (korvaa arp-komennon ja tarjoaa vastaavan toiminnallisuuden myös IPv6:een)
- Naapuritaulussa (IPv4:ssä myös ARP-taulu, sanoista Address Resolution Protocol) on tieto siitä, mikä IP vastaa mitäkin hardware-osoitetta ja interfacea; päivittyy yleensä automaattisesti
- Naapuritaulua voi tutkia ja muokata, esim (optioita on paljon enemmänkin):

  ip neigh
  - Tulostaa naapuritaulun

  ip neigh add *ip-address hw-address* dev *device*
  - *Lisää naapuritauluun rivin*

  ip neigh change ...

  ip neigh replace...

  ip neigh delete ...
- ARP Proxy: kone reagoi toiselle (aliverkon ulkopuolella olevalle) koneelle tarkoitettuihin paketteihin (ja ohjaa ne jotain tunnelia pitkin perille); harvoin tarpeen, vaikea käyttää turvallisesti

virsh start/shutdown/...

- Virtuaalikoneen sammutus/käynnistys: virsh...
  - start (sammutetun koneen käynnistys)
  - shutdown (”siisti” sammutus, edellyttää acpid'ia)
  - destroy (sammutus virtanapista)
  - reset (reset-nappi, destroy + start)
  - suspend (tilapäinen pysäytys)
  - resume (herätys suspend-tilasta)
  - dompmsuspend (suspend kutsuen vm:n käyttöjärjestelmän virranhallintaa)
  - dompmwakeup (herätys dompmsuspend-tilasta)
  - save (~läppärin hibernate, muisti & koneen tila tiedostoon)
    - Tilatiedosto oletuksena vain rootin luettavissa
  - restore (palautus save-tilasta)
  - managedsave (tilapäinen/sisäinen save, jatko start-komennolla, ei restore)
  - autostart (käynnistys automaattisesti kun alustakone bootataan)
  - dumpxml (tulosta koneen määrittely xml-muodossa)
  - undefine (poista koneen määrittely järjestelmästä)
  - define (palauta koneen määrittely xml-tiedostosta)

Offline migration

- Virtuaalikoneen siirto alustakoneesta toiseen sammuttaen se ensin (ikäänkuin kovalevyn siirto fyysisestä koneesta toiseen):
  - Nykyisessä alustakoneessa (esim. lonka7):

    virsh shutdown kone1

    virsh dumpxml kone1 > kone1.xml

    virsh undefine kone1

    scp kone.xml kone1.img [kone1b.img...] lonka6:
  - Uudessa alustakoneessa (lonka6):

    virsh define kone1.xml # editoi ensin jos levyn polku vaihtui
- Jos alustakoneet eivät ole samassa aliverkossa virtuaalikoneen verkkokonfiguraatio pitää muuttaa, useimmiten helpointa tehdä ennen siirtoa; yleensä edellyttää vastaavaa muutosta myös nimipalveluun ja palomuureihin
- Joe levytiedoston polku vaihtuu, riittää kun sen muuttaa xml-tiedostoon

Pseudolive migration

- Virtuaalikoneen siirto boottaamatta (”levyhibernaation” kautta, kuten läppärille hibernate ja siirto toiseen paikkaan):
  - Nykyisessä alustakoneessa (esim. lonka7):

    touch kone1.save # hakki root-oikeuksien välttämiseksi, ei tarpeen jos root/sudo

    virsh save kone1 kone1.save

    virsh dumpxml kone1 > kone1.xml

    virsh undefine kone1

    scp kone1.xml kone1.save kone1.img [...] lonka6:
  - Uudessa alustakoneessa (lonka6):

    virsh restore [--xml kone1.xml] kone1.save
- Jos alustakoneilla jaettu levy, scp ja touch-temppu ei tarpeen
- Verkkokonfiguraation ja levypolun vaihto tarvittaessa kuten edellä (jos xml-tiedosto muuttunut tarvitaan lisäksi --xml -optio)

Live migration

- Virtuaalikoneen siirto ”livenä”, kone pysyy käynnissä vanhassa alustakoneessa kunnes siirto on valmis, vain minimaalinen katko kun viimeksi käytössä ollut muistin osa siirretään ja verkkoyhteys vaihtuu
- Edellyttää jaettua levyä (NFS, iSCSI tms), levyimagen pitää näkyä samassa polussa molempiin koneisiin
- Tiedonsiirtovaihtoehtoja useita, esim. ssh:

  virsh migrate --live kone qemu+ssh://alustakone/system
- Yleensä edellyttää root-oikeuksia alustakoneisiin
- Vanhan ja uuden alustakoneen täytyy olla samassa aliverkossa, virtuaalikoneen verkko- tai levykonfiguraatiota ei voi muuttaa

nginx

- ”Engine X”, toiseksi yleisin (apachen jälkeen) www-palvelin Linux-ympäristössä, yleistyy nopeasti
- Ei yhtä ”full-featured” kuin apache, mutta monissa tilanteissa kevyempi (etenkin muistin tarve pienempi, joskus vertailukelpoinen jopa lighttpd:n kanssa)
- Yleinen (tehokas) edustapalvelimena (proxy)
- Ubuntussa tai Debianissa asennus onnistuu tavanomaiseen tapaan (apt-get install nginx), mutta valmiiksi paketoitu versio on usein vanha. Jos halutaan uudempi tehdään ensin

  sudo add-apt-repository ppa:nginx/stable

  - Jos add-apt-repository -komentoa ei löydy: apt-get install software-properties-common
    - Repositorymääritys menee hakemistoon /etc/apt/sources.list.d
  - Jos jo asennettu vanhempi versio, voidaan päivittää tekemällä (edellisen jälkeen)

    apt-get update; apt-get upgrade nginx # tai apt-get dist-upgrade, päivittää kaiken muunkin

nginx konfiguraatiotiedostot

- Konfiguraatiohakemisto /etc/nginx
- Pääkonfiguraatiotiedosto /etc/nginx/nginx.conf
- Lisäkonfiguraatiotiedostoja /etc/nginx/conf.d/*.conf (automaattisia)

  ja /etc/nginx/snippets/*.conf (otettava käyttöön erikseen)
- Omia konfiguraatiotiedostoja voi sisällyttää muualtakin include-direktiivillä
- Määritellyt virtuaalihostit (www-) hakemistossa /etc/nginx/sites-available, käytössä olevat /etc/nginx/sites-enabled (linkkejä edelliseen)
- Oletus /etc/nginx/sites-available/default, sijainti:

  grep root /etc/nginx/sites-available/default
  - /var/www/html, /usr/share/nginx/html tms

  grep index /etc/nginx/sites-available/default
  - Jos useita vaihtoehtoja, ensimmäistä löytynyttä käytetään
  - Yleensä ainakin index.html toimii
- Jos default-konfiguraatiota ei käytetä:

  rm /etc/nginx/sites-enabled/default

nginx: virtual host, esimerkki

```
cat >/etc/nginx/sites-available/tt2.student.it.jyu.fi<<\EOF
server {
  listen 80; # Ipv4; voisi olla myös esim. 8080 tai 443 (https)
  listen [::]:80; # IPv6
  server_name tt2.student.it.jyu.fi;
  root /var/www/tt2.student.it.jyu.fi/html;
  index index.html;
  location / {
      try_files $uri $uri/ =404;
   }
}
EOF
ln -s  ../sites-available/tt2.student.it.jyu.fi /etc/nginx/sites-enabled
mkdir -p /var/www/tt2.student.it.jyu.fi/html
nano /var/www/tt2.student.it.jyu.fi/html/index.html # jotain sopivaa
service nginx reload
tail /var/log/nginx/error.log # jos ei toimi...
```

nginx konfiguraatiodirektiivit

- Nginx'in konfiguraatiotiedostossa olevat määritykset (direktiivit) voivat olla globaaleja tai jossain aaltosuluilla {} rajatussa kontekstissa: http{} server{} location{} events{}
- Direktiivien loppuun tulee yleensä puolipiste (ei }:n jälkeen)
- Yleisiä globaaleja asetuksia: user (käyttäjätunnus jolla nginx pyörii), worker_processes (montako palvelinprosessia käynnistetään), pid (minne nginx-pääprosessin PID talletetaan); näitä ei yleensä tarvitse muuttaa
- Konfiguraatiotiedostoon voi lukea muita include-direktiivillä, jokerit toimivat
- Kaikki http(s)-palvelimiin liittyvät direktiivit (erityisesti myös server{}) ovat http{}:n sisällä; kaikki .../sites-enabled/... tiedostot tulevat automaattisesti sen sisään (toteutettu include-direktiivillä html{}:n sisällä)
- Virtual host -määritykset tehdään server{} -direktiivillä, niiden sisällön ohjaukset sen sisällä location{}'illa
- events{} ohjaa yhteyksien muodostumista (esim. worker_connections: kuinka monta yhtaikaista yhteyttä per palvelinprosessi sallitaan)

nginx: server{}

listen [*osoite*][:*portti*] [*optiot*];

- jos :*portti* puuttuu, oletetaan 80
- jos *osoite* puuttuu, oletus = * = kaikki IPv4-osoitteet
- [::] = kaikki IPv6 -osoitteet tai myös IPv4 optiolla ipv6only=off

server_name *nimi* [*nimi...*];

- nimi tai nimet joihin vastataan; ensimmäinen ensisijainen
- jokerit ja regexp'it sallittuja: *.example.com ~^www\..*\.example\.net

root *polku*;

- sisällön juurihakemisto, johon suhteessa URL-polut tulkitaan

index *file* [*file...*];

- tiedostonimi jota käytetään URLin viitatessa hakemistoon
- jos useita, käytetään ensimmäistä joka löytyy

error_log *file* | stderr | syslog:server=*...*

- minne virheilmoitukset kirjoitetaan (paljon optioita)

location ...

nginx: location{}

location [= ~ ~* ^~] *URI* {...}

location @*nimi* {...}

- URIsta riippuvia asetuksia (server{}in sisällä); voi olla useita sisäkkäin
- URI voi olla

  polun *alku* (esim. /); jos edessä ^~, ohittaa regexpit
  - käsitellään pituusjärjestyksessä, pisin voittaa

  =koko polku (”= /kala/hauki.html”)
  - voittaa kaikki muut; voi olla esim. yhtaikaa sekä ”/” että ”= /”

  ~regexp (”~ \.php$”), ~* case-insensitive
  - regexp'it voittavat polunalkumääritykset ilman ^~ -prefiksiä
  - regexp'it tulkitaan järjestyksessä, ensimmäinen voittaa
- @*nimi* määrittää nimetyn sijainnin käytettäväksi uudelleenohjausviittauksissa
- location{}in sisällä voi olla mm.

  root, index, error_log, proxy_pass ...

nginx: try_files

try_files *file ... URI;*

- Vaihtoehtoja miten tiedosto tulkitaan, ensimmäinen löytynyt voittaa
  - $uri on alkuperäinen polku sellaisenaan
  - =*code* palauttaa virhekoodin
- Esim.

```
location / {
  try_files /huoltokatko.html $uri $uri/ $uri/index.html $uri.html =404;
}
location /pictures/ {
  try_files $uri /pictures/oletus.jpg;
}
location / {
  try_files $uri @varakone;
}
location @varakone {
  proxy_pass http://192.168.6.7:8080;
}
```

nginx: php

- sudo apt-get install php5-common php5-cli php5-fpm
- Haluttuun konfiguraatiotiedostoon:

```
location ~ \.php$ {
  fastcgi_split_path_info ^(.+\.php)(/.+)$;
  try_files $fastcgi_script_name =404;
  set $path_info $fastcgi_path_info;
  fastcgi_param PATH_INFO $path_info;
  fastcgi_pass unix:/var/run/php5-fpm.sock;
  fastcgi_index index.php;
  include fastcgi.conf;
}
```

- sudo service nginx restart

nginx: rewrite

rewrite *regexp korvaus* [*lippu*]

- Muuntaa URIn toiseksi regexp-säännöllä
- Jos useita, suoritetaan järjestyksessä
- Liput:
  - break: lopeta rewrite-sääntöjen suoritus tähän
  - last: lopeta tähän ja ala käsitellä muutettua URIa alusta
    - voi johtaa silmukkaan...
  - redirect: palauta ”temporary redirect” -koodi
  - permanent: palauta ”permanent redirect” -koodi
- Esim.

```
location /download/ {
  rewrite ^(/download/.*)/media/(.*)\..*$ $1/mp3/$2.mp3 break;
  rewrite ^(/download/.*)/audio/(.*)\..*$ $1/mp3/$2.ra  break;
  return  403;
}
```

nginx: proxy_pass

proxy_pass *URL*;

- Ohjaa (yleensä location{}in määräämän) URLin toiseen
- Esim.

```
location / {
  proxy_pass http://192.168.122.4:8080/;
}
location /images/ {
  proxy_pass http://192.168.122.5/pics/;
}
```

- Yhdistettävissä esim. rewrite-direktiivin kanssa:

```
location /user/ {
    rewrite    /user/([^/]+) /users?name=$1 break;
    proxy_pass http://192.168.7.8/;
}
```

nginx: reverse proxy

- Esimerkki: halutaan erottaa virtual hosteja sisäisesti eri virtuaalikoneisiin.
- Ohjataan kaikki ulkoa tulevat pyynnöt edustapalvelimena toimivalle nginx'ille, jossa on kullekin virtual hostille tämäntapainen määritys:

```
server {
  listen :80;
  server_name: 1.example.org;
  location / { proxy_pass http://kone1:8080/; }
}
```

- Virtual hosteja voidaan jakaa virtuaalikoneisiin kuten halutaan, jossain voi pyöriä nginx, jossain apache, jossain lighttpd... jopa useita samassa koneessa eri porteissa, edustakoneen huolehtiessa ohjauksesta oikeaan paikkaan

umask

- Shellin (bash) sisäinen komento: asettaa oletusarvon luotavien tiedostojen oikeuksille
- Vanha syntaksi oktaalinen bittimaski **poistettaville** oikeuksille, esim.

  umask 022 # ryhmältä ja maailmalta w-oikeus pois

  umask 067 # ryhmältä rw pois, maailmalta kaikki pois
- Uusi syntaksi **sallitut** oikeudet kuten chmod'issa, esim. ylläolevat toisin esitettynä:

  umask u=rwx,g=rx,o=rx

  umask u=rwx,g=x,o=
- Ilman argumentteja tulostaa voimassaolevan asetuksen, oletuksena oktaalimuodossa, optiolla -S symbolisena