Terhelés-elosztás és a hálózat

2017. február 10.

Illeszkedés a sorozatunkba

- A számítógép-hálózatok már régóta szinte mindennek az alapjai
- Ebből fakadóan néhány felmerülő feladat nem egyértelműen a hálózatüzemeltető feladata
- De mégis, a megoldás gyakran
  - ✓ egyszerűbb
  - ✓ üzembiztosabb
  - ✓ skálázhatóbb
  - ✓ több előnnyel jár

  amennyiben a hálózati rétegben történik

- Ebben a kontextusban a rétegződést nem a klasszikus hét rétegű OSI rétegként kell értelmezni

Alkalmazás-üzemeltetés
Szerver-üzemeltetés
Hálózat-üzemeltetés

Frontend réteg (web)
Alkalmazás réteg (app)
Adatbázis réteg (db)

- Miért foglalkozik a hálózatos mérnök mással, mint a hálózat? Példákért nem kell messzire menni:
  - ➔ Majdnem az egész előző oktatás erről szólt
  - ➔ De ilyen példa lehet a címfordítás is: elköltöző szerver, migrálandó alkalmazás

Ismerős a láthatáron

- Honnan indultunk?
- Keretek, csomagok, fejlécek, kábelek, rétegek
- Vlanok, feszítőfa, bpdu, trunking, interfész jellemzők
- Mindezek és kapcsolódo technológiák, szabványok részletes vizsgálata, elemzése, tanulmányozása
- Ezek a technikák meglehetősen specifikusak, cél-orientáltak, hardverhez kötöttek, részfeladatokat oldanak meg
- Az implementáció csak hálózatra jellemző, kívülről láthatatlan dolgokra fókuszál amiből a hálózaton kívüli emberek (szerver- és alkalmazás-üzemeltetők) semmit nem látnak, nem tudnak
- A hálózatot azonban néha be kell, néha be javasolt vonni magasabb szintű problémák megoldásába is
- Ezek a feladatok viszont komplexek, egyediek, nem feltétlenül szabványokra alapuló megoldást kell találni
- Specializált, protokollokra épülő célszoftverrel nem minden esetben lehet megoldást adni a hálózat keretein belül
- Szükség van tehát valamire, ami flexibilis, sokoldalú, alakítható a pillanatnyi igények ismeretében

- A hálózati világ is felfedezte magának a linux alapú megoldásokat

Kitekintő

Visit our blog for quick and short network tutorials! For free online trainings and seminars, follow us, or visit our website: http://svs.cx.

Smart Vision Solutions ltd.
Network consulting and implementation tailored to you

Mai téma

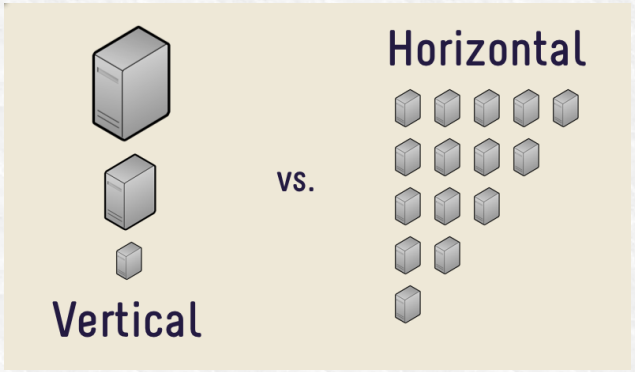
- Alapvetően nagyteljesítményű szolgáltatásokhoz tervezett hálózatokra fókuszálunk
- Ezek a kihívások nem elsősorban iskolai körülmények között fordulnak elő
- Mégis hasznos a témában elmerülnie mindenkinek, aki a jövőben hálózatokkal kíván foglalkozni

- A szolgáltatás bármilyen alkalmazás lehet: streaming, weboldal, levelezés, bármi
- Tulajdonképpen a hálózat szempontjából teljesen mindegy, hogy mi a szolgáltatás
- Az egyszerűség kedvéért ma egy weboldal megjelenítése a laborban a “szolgáltatás”

- Mi szükség van nagy teljesítményhez hálózatot tervezni? Mi köze egyátalán a hálózatnak ehhez?

- Tudjon egy szerver i3-mal 1 GB Rammal, HDD-vel kiszolgálni 50 oldallekérést másodpercenként
- Mi történik, ha érkezik 100/mp forgalom? Mondjuk RAM bővítés 2GB-re
- Ha 150/mp? Mondjuk i3-ról erősebb CPU-ra kell váltani.
- Ha 250/mp? Mondjuk SSD-re váltani és még több RAM, még több CPU.
- Mi történik, ha elérte a bővíthetőség határát a szerver és még mindig nő a forgalom? Erősebb szerver.
- Mi történik, ha a piacon rentábilisan elérhető max kapacitás mellett is tovább nő a forgalom?
- Az ész nélküli bővítés nem vezet sehova, koncepciót kell váltani.

- Ez az új koncepció a terhelés-elosztás, load balancing
- Ha a **szerverek bővítése** a **vertikális** növekedés, akkor a **terhelés-elosztás** a **horizontális** növekedés



- Hogyan működik egy ilyen rendszer, milyen lehetőségek vannak?
- Milyen problémákkal találkozunk, ha ilyen rendszer alá tevezünk hálózatot?

A reptéri parkolójárat példája

- A skálázhatóság nem csak a hálózatokban jelentkező kihívás
- London Heathrow reptere Európa legnagyobbjainak egyike, évi 75 millió utas fordul meg itt (Budapest: évi kb 11 millió)
- Az utasok egy része a reptéren parkol, ahonnan valahogy el kell jutnia a terminálhoz
- Ha 10 percenként 50 fős busz szállítja az utasokat, akkor **egy órában** 50 * 6 = **300 utast** tudunk a terminálhoz vinni
- Az éjszakai repülési tilalom miatt kb egy nap 18 órában akarnak utasok a terminálhoz jutni, azaz **napi** 18 * 300 = **5 400 utas** a max kapacitás egy nap. Heathrow egy nap kb. 205 000 utast szolgál ki
- Heathrow bevezette a **pod**-okat, amelyek önvezető kis járgányok, egyszerre négy embert tudnak szállítani. Az út velük a parkolóba öt perc, míg hagyományos busszal kb. 10-15 perc volt. A buszra átlagosan 10 percet kellett várni, a pod-ra harminc másodpercet.
- Egy pod egy utat 5.5 perc alatt tesz meg, óránként tízszer tud fordulni, azaz kapacitása 4 * 10 = 40 utas
- 21 pod van, tehát a rendszer kapacitása 21 * 40 = **840 utas óránként**, 18 órás **nap** alatt **15 120 utas**
- A podok egyébként három másodpercenként követhetik egymást, tehát még bőven van tartalék
- A fenti teljesen hétköznapi példa egy modell a horizontális skálázhatóságra

- Felejtsük el egy pillanatra a kapacitást, gondoljunk a rendszer hibatűrésére: mi történik ha elromlik egy busz? 50 ember várakozik 10 percet a következő buszra, amire nem fér fel.
- Mi történik ha egy pod romlik el? Négy ember várakozik sokkal kevesebbet, mint 10 perc.
- A buszok és podok megfelelői a szerverek: ha egy szerver elromlik, mennyi idő azt pótolni?

Bővítés: H vagy V?

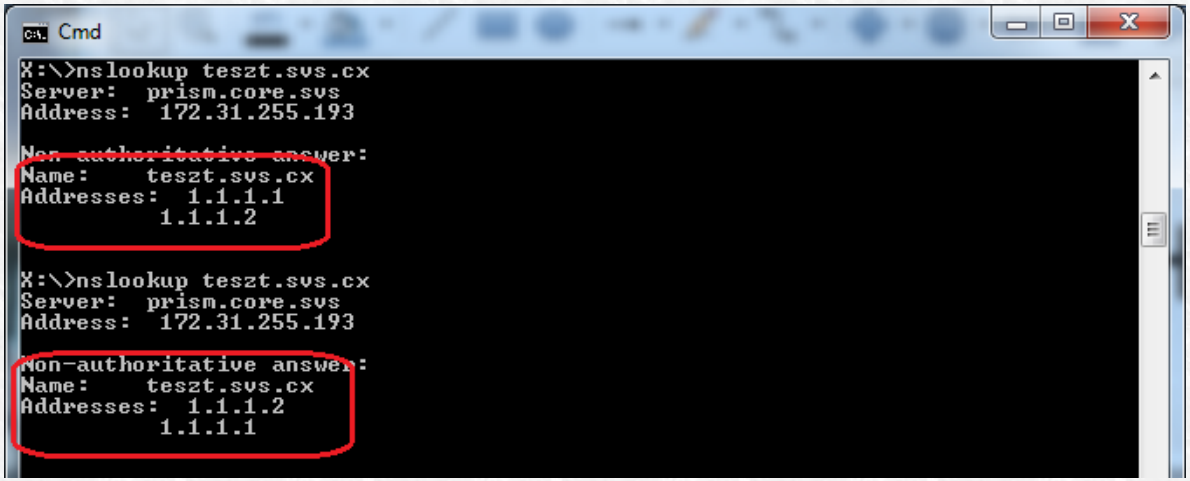
- A terheléssel tehát kell foglalkozni, előbb utóbb mindenből skálázhatósági probléma lesz
- Nincs univerzális recept, minden esetet egyedileg kell megítélni

Szempont	Vertikális bővítés	Horizontális bővítés
Hol a határ?	Minden piacon van legnagyobb, legdrágább alkatrész, de ha azt is megvetted, nincs tovább. Ezt a korlátot hamar el lehet érni.	Bármiből lehet mégegyet venni és üzembe állítani. A korlát itt hely, áram, IP cím igény, amit nehezebb elérni, de ha bele is ütközöl, könnyebb megoldani, mint várni egy még nem létező alkaltrészre.
Mennyibe kerül?	Önmagában egy-egy alkatrész olcsóbb lehet, mint teljes új szervert venni, még ha régi, olcsóbb szerverről is van szó.	Egy régebbi szerver beszerzése lehet olcsóbb, mint egy újabbé, de drágább, mintha csak alkatrészeket kellene venni.
TCO?	Minden bővítés leállással jár. Üzemeltető személyzet több kell, emelt mennyiségű túlóra várható, minden karbantartás éjjel történik.	N-szer annyi szerver kevesebb, mint N-szer annyiba kerül, bővítések csak teljesítmény-csökkenéssel járnak, bármikor elvégezhetőek, kisebb létszámú üzemeltető elegendő.
Kompexitás?	Túl sok ész nem kell hozzá, az egyetlen kihívás olyan alkatrészt venni, ami kompatibilis a meglévőkkel.	Tervezést igényel. Mind ez, mint az üzemeltetés az átlagnál több szakértelmet kívánhat. Nagyobb mennyiség esetén új skálázási problémák lépnek fel.

Nyilván ma itt most a horizontális bővítést vizsgáljuk, illetve konfiguráljuk be.

DNS round robin

- A hátralévő időben terhelés-elosztó megoldásokat nézünk át, illetve vizsgálunk meg
- A legegyszerűbb terhelés elosztáshoz nincs szükség extra hardverre, megvalósítható DNS-ből is
- Ezt leggyakrabban egyszerűen ugyanahhoz a névhez tartozó több A rekorddal lehet elérni
- A kliens sem feltétlenül fordul hiba esetén a másik kapott címhez, így lehet, a semmibe küldöd a forgalmat



DNS round robin (2)

Előnyök:

- Ha már megvan a második szerver, percek alatt implementálható
- Nagyjából elronthatatlanul egyszerű
- Egyszerre globális és loklális terhelés-elosztás
- Nincs SPoF a terhelés-elosztó ponton
- Ingyen van

Hátrányok:

- A DNS cache ellened dolgozik
- Nincs ráhatásod a konkrét sorrendre, sem a pontos terhelés megosztásra
- A TTL értékek kliens oldali, illetve köztes DNS szerver oldali kezelésére nincs garancia
- A DNS koncepciója túl merev, minden változás terjedéséhez idő kell
- A DNS szerver a valódi szerver hibája esetén is visszaadja az A rekordot
- Nem egyenletes terhelés a valódi szervereken

DNS RR++ = Cisco GSS

A logikai és egyben evolúciós sorban következő lépés az intelligens DNS szerver

- Az előző megoldás hibái közül néhányat egyszerűen meg lehetne oldani

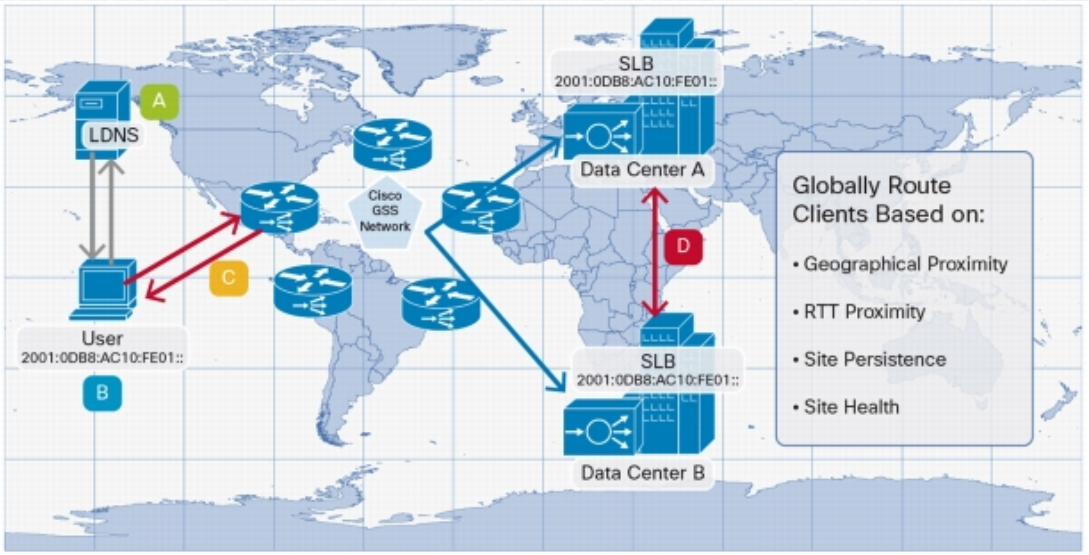
```
#!/bin/bash

ping -c 1 -w 1 -q 1.2.3.4 2>/dev/null

if [ $? -ne 0 ]; then
   echo "update dns set valid=0 where host='dummy.demo' and ip='1.2.3.4';" | mysql dns
fi
```

- A fentiekhez egy GUI-t és redundáns működést adva el is jutottunk a Cisco GSS-hez
- GSS: Global Site Selector – egy DNS szerver
- Azaz kettő, mert minimum párban telepítendő
- A GSS folyamatosan...
  - ... monitoroz
  - ... ellenőriz
  - ... nyilvántart
  - ... válaszol a megfelelő adattal

- A végpontok mindig csak a számukra megfelelő IP címet kapják meg



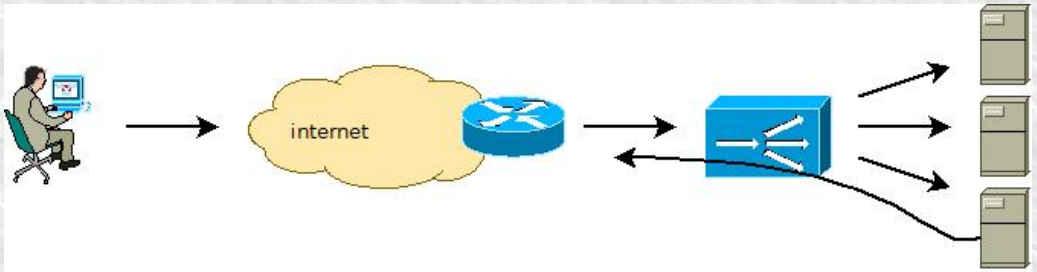
GSS++ = LB

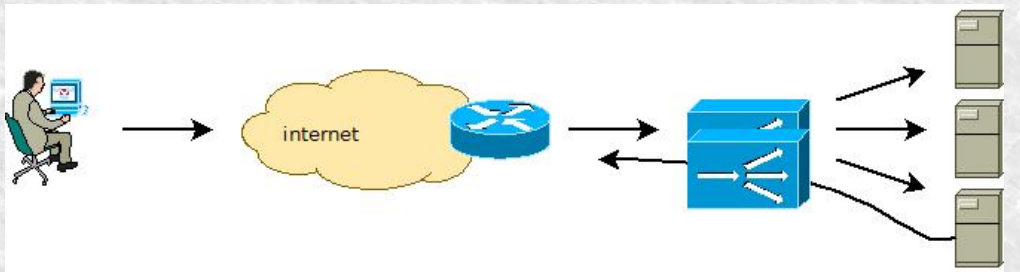
- Az eddigi megoldások közös jellemzője, hogy a DNS-re épített mindegyik
- Van amit megold a dobozos termék, van amit le lehet scriptelni, de a DNS nem terhelés-elosztásra van tervezve
- A terhelés-elosztáshoz tehát megpróbáltuk a DNS-t, ami nem tökéletes. Valami új kell.
- Ez az új dolog a Load Balancer ami konkrétan a terhelés elosztására szolgál
- A Cisco-nak is van egy egész termékcsaládja: Application Networking néven
- CSS: Content Services Switch, CSM: Content Switching Module, ACE: Application Control Engine
- Van viszont egy egész cég, aki terhelés-elosztásra specializálódott: az F5
- Számunkra ma érdekes termékei az LTM (Local Traffic Manager) és a GTM (Global Traffic Manager)
- Ma este F5 LTM-et fogunk telepíteni, konfigurálni és tanulmányozni működés közben
- De előbb az elméletet kell tisztáznunk!

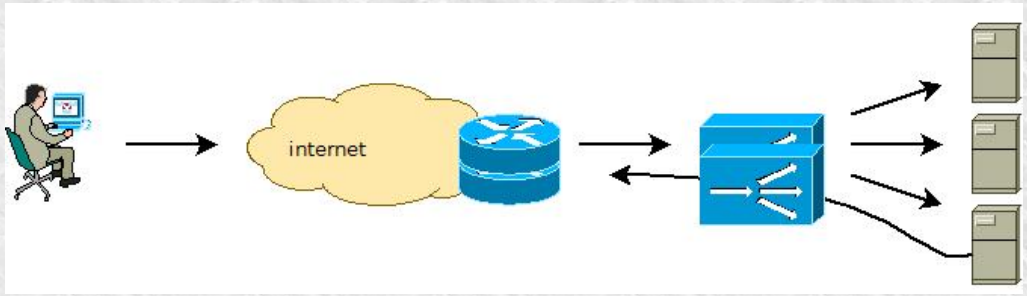
LB vs. a barkács linuxok

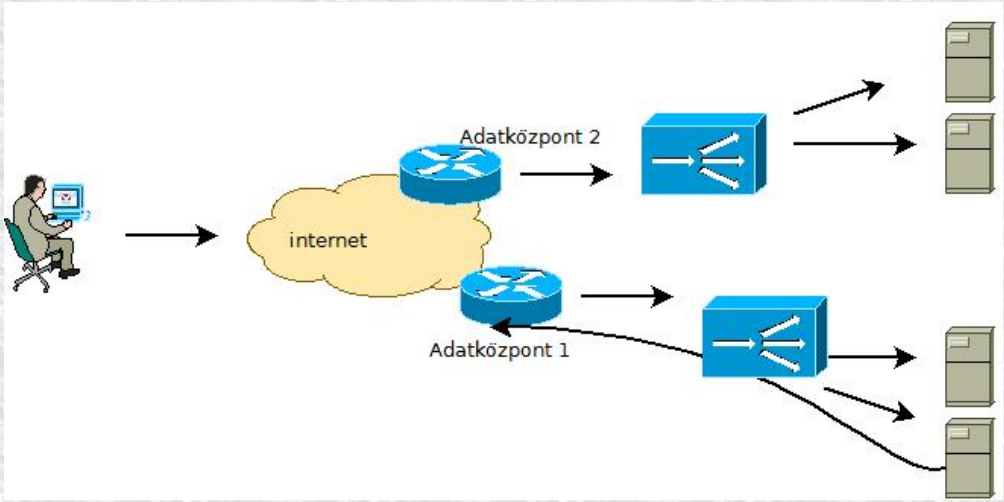
- A LB koncepciója szerint a forgalmat a DNS egy úgynevezett **VIP**-re irányítja (Virtual IP)
- A VIP nem a szerver IP címe, ahol a tartalom van, hanem egy cím a LB eszközön
- Az LB van beállítva úgy, hogy az erre a címre érkező forgalmat tovább küldje a valódi szervereknek (**real server**)
- A kliens a valódi szerverekkel közvetlen kapcsolatba nem tud kerülni és a létezésükről sem tud, csak a VIPet látja
- Ebből következően a LB funkciói transzparensek a látogató számára
- Miben különbözik egy LB egy sima linuxos előtét szervertől vagy reverse proxytól? Nem sokban. A jellemzők:
- Konfigurálható GUI-n, vagy CLI-n át (a gyártó saját CLI-t készít a célra saját parancsokkal)
- Teljesítménye sokszorosa egy sima PC-nek
  - CSS: 6 – 40 Gbps
  - CSM: 165 000 új kapcsolat/mp, legfeljebb 4000 VIP-en fogadva, legfeljebb 16000 valódi szervernek továbbítva
  - ACE: 4 – 16Gbps, legfeljebb 325 000 új kapcsolat / mp, 15 000 SSL új kapcsolat/mp
- SSL **offloading**: a titkosítás erőforrás igényes, az LB elvégzi a szerverek helyett
- TCP offloading: a TCP kapcsolatot is az LB építi fel, a valódi szerverekhez csak a hasznos kapcsolatok kerülnek
- Protokoll-ismeret, protokoll-manipuláció (HTTP fejlécek beszúrása, átírása)
- Telepítésre kerülhet routed vagy bridged üzemmódban is (L2 vagy L3 üzemmódban)
  - Routed (L3): amikor a kliens és a szerver különböző IP tartományokban van, az LB routing hop a kettő között
  - Bridged (L2): amikor a kliens és a szerver azonos IP tartományban van, az LB bridge lesz kettejük között
- Folyamatos teljesítmény-monitorozás (ping, http oldallekérés, tetszőleges tcp port ellenőrzés, scriptek futtatása)
- A monitorozás terminológiája a **probe**, vagy a **monitor**
- A valódi szervereket csoportosítva kezeli (**serverfarm**, **pool**)
- Ha a protokoll igényli, egy kapcsolathoz tartozó minden kérés ugyanarra a szerverre kerül (**persistence**, **stickyness**)
- A kiemelt protokollok (http, ftp, dns, rtsp, radius, rdp) mellett tetszőleges alkalmazást / portot tud irányítani
- A kiemelt protokollok esetén IPS, IDS funkció, protokoll-analizálás / védelem
- Többszintű adminisztrációs lehetőség (delegált üzemeltető korlátozott jogokkal)
- Virtualizáció: egy fizikai LB-ben lehetőség több önálló, izolált LB megvalósítására amik egymástól függetlenek
- Valamint páros telepítés esetén teljes leállás esetén is a végfelhasználó számára transzparens átállás a tartalékra (**HA**)

LB vs SPoF

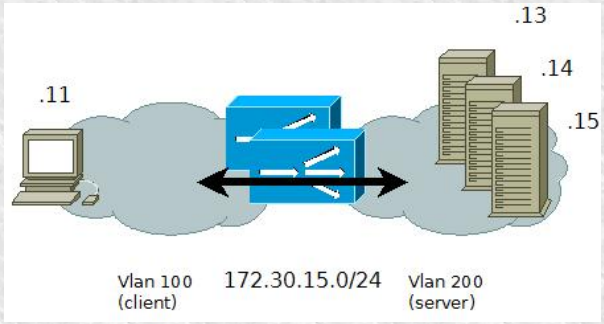


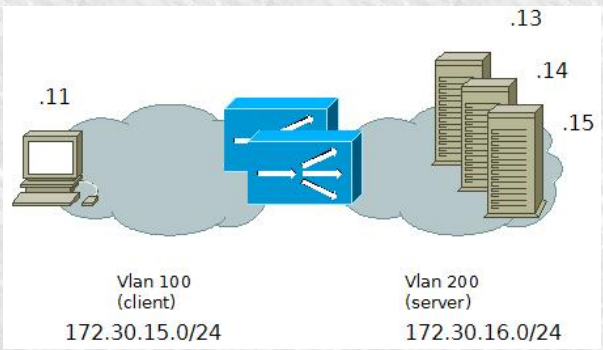


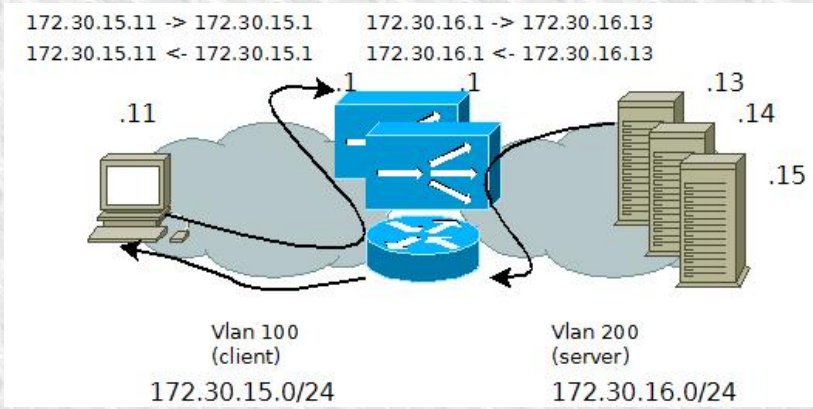


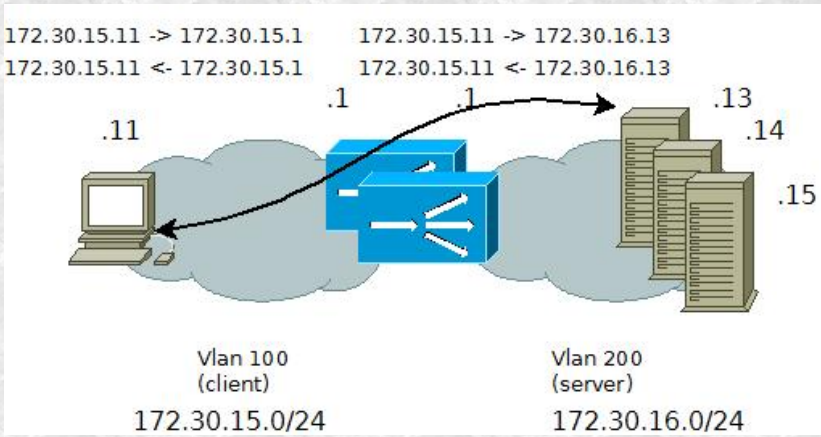


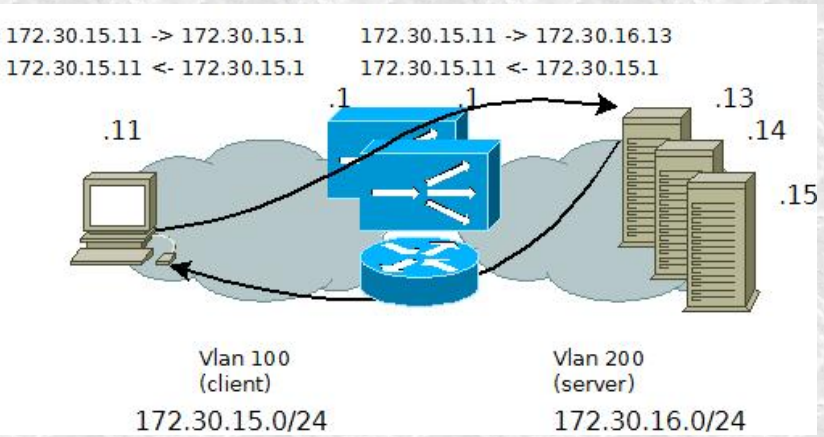
LB modellek





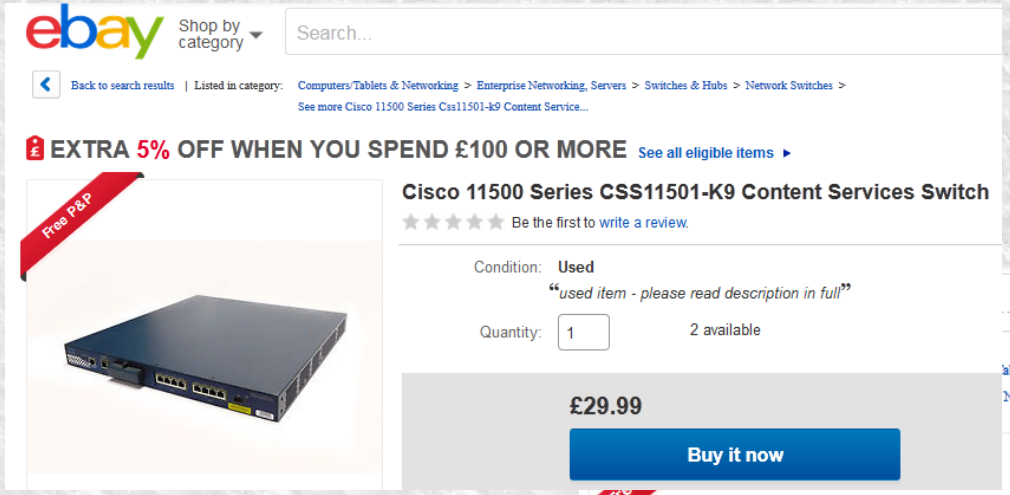


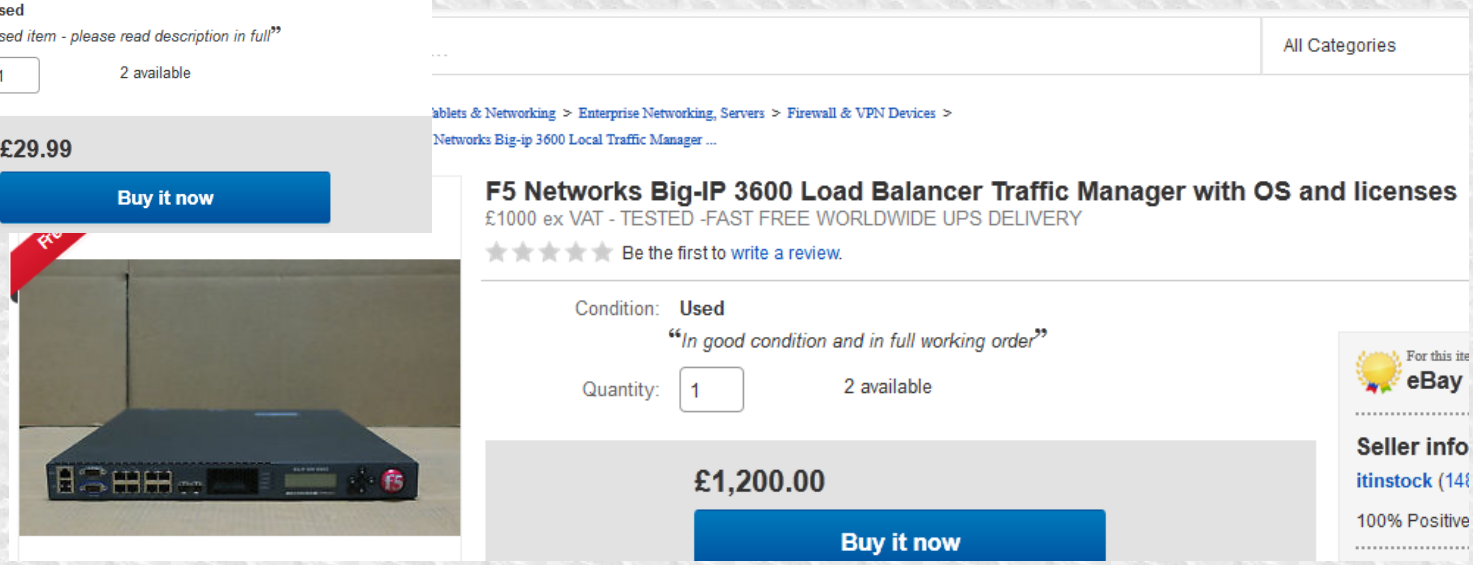




Mennyiből?

- Lehet nekiállni építeni sima PC-ből és linuxból, a feladat ismert, végülis mindent le lehet scriptelni...
- Lehet kukázni használtat olcsón
- Lehet milliókat költeni (ez nem csak a vas, hanem: licensz, mérnökóra, hálózat/hosting, support egyben több eszközre)
- Minden attól függ: mekkora forgalomra, mennyire redundánsat, milyen funkciókkal





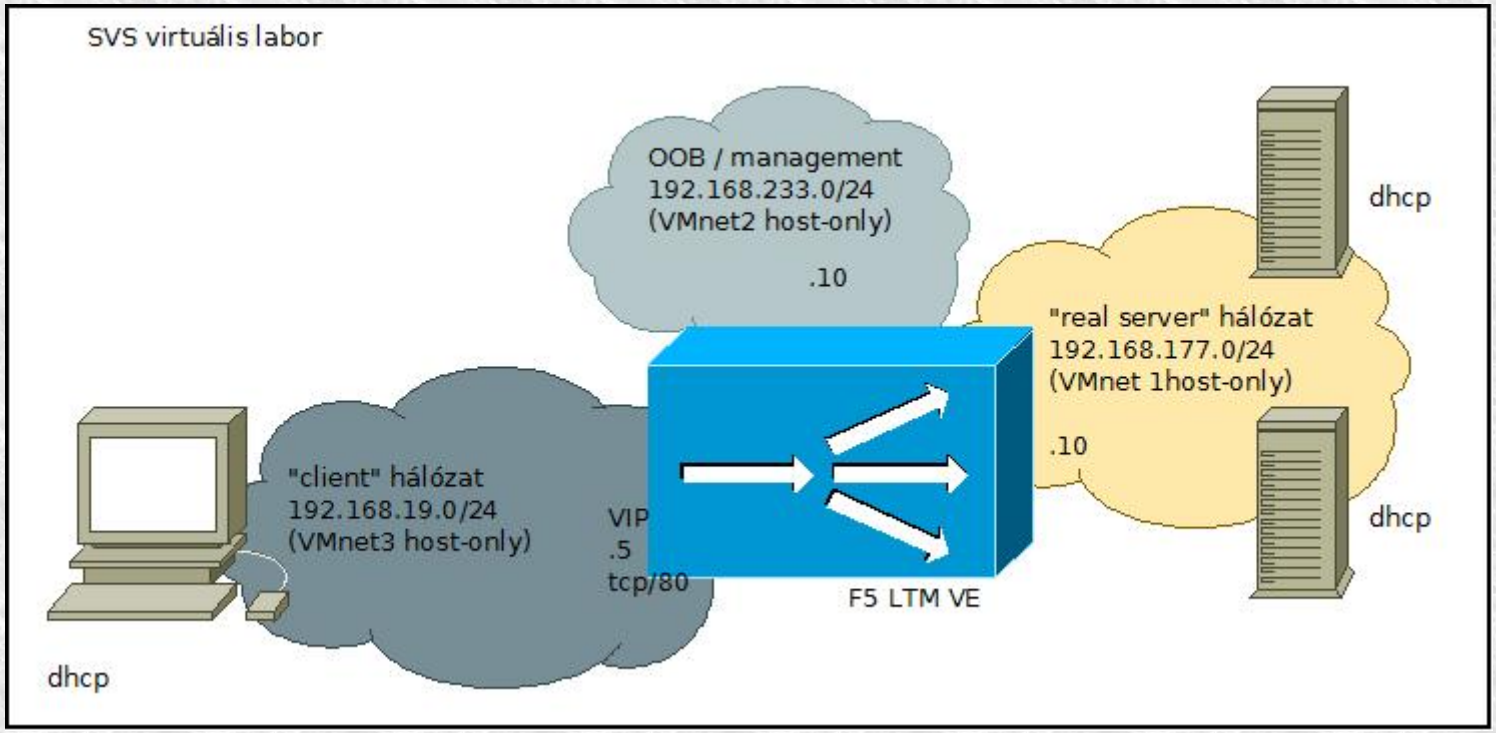
Visit our blog for quick and short network tutorials! For free online trainings and seminars, follow us, or visit our website: http://svs.cx.

F5 LTM

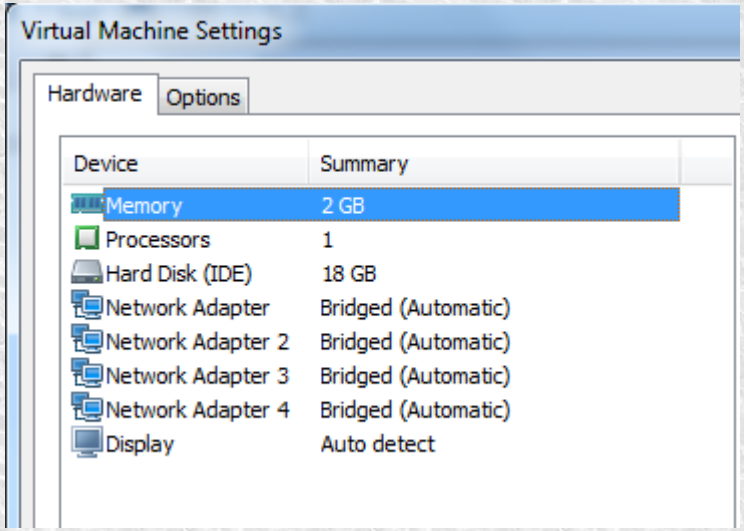
- Az F5 jelenleg a piacvezetők egyike terhelés-elosztás terén, a vezető termékeik az LTM és a GTM
- Az LTM maga egy LB, megfeleltethető a Cisco CSS/CSM/ACE hármasnak
- A GTM egy DNS szerver, megfeleltethető a Cisco GSS-nek
- Linux alapú és nincs is nagyon elrejtve a shell: aki linuxban otthonosan mozog, itt sem fog eltévedni
- Alapvetően azonban GUI-ról vezérelt
- Tudásában és teljesítményében messze megelőzi a Cisco termékeket, így nyilván ezzel foglalkozunk inkább

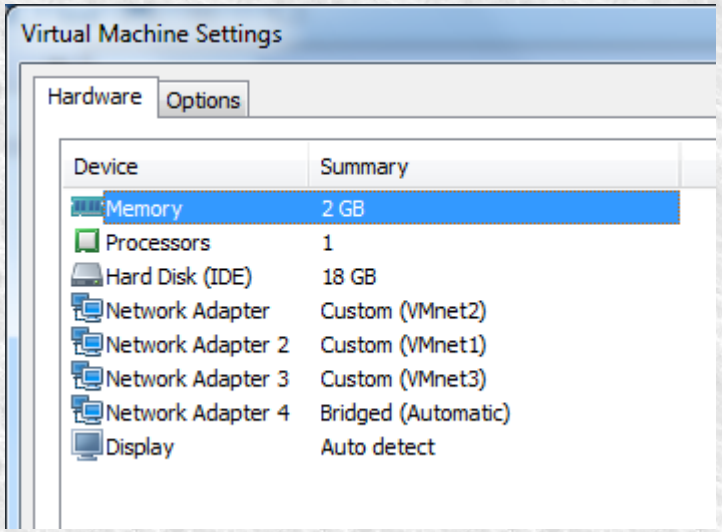
- Ma este:
  - Felinstallálunk egy F5 LTM-et
  - Van készen két linuxunk, rajtuk egy-egy webszerverrel
    - A linuxok installálását nem részletezzük, ez nem linux tanfolyam, mindenki képes egy linuxot feltenni
    - Webszervert szintén bárki a kedvenc disztribúciója alá fel tud tenni, nem kell hozzá előadás
  - Beállítunk különféle LB metódusokat, megnézzük mi történik a hálózaton
  - Néhány hibát ejtünk a rendszeren, megnézzük mit lát a végfelhasználó
  - Megnézzük, hogyan változik meg a hálózatüzemeltető élete, ha terhelés-elosztók kerülnek a hálózatába

Mai labor

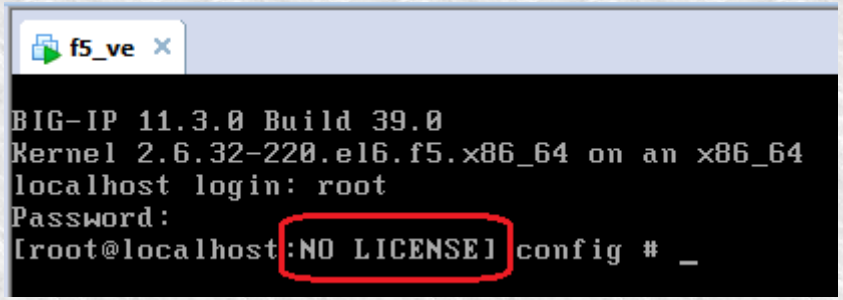


Előkészületek





- Nekünk most csak három interfészre van szükségünk (oob, client, server)
- A fenti listában az interfészek sorrendje fontos, a virtuális gép ebben a sorrendben látja azokat
- konzol login: root / default
- Első tennivaló: licenszelés!



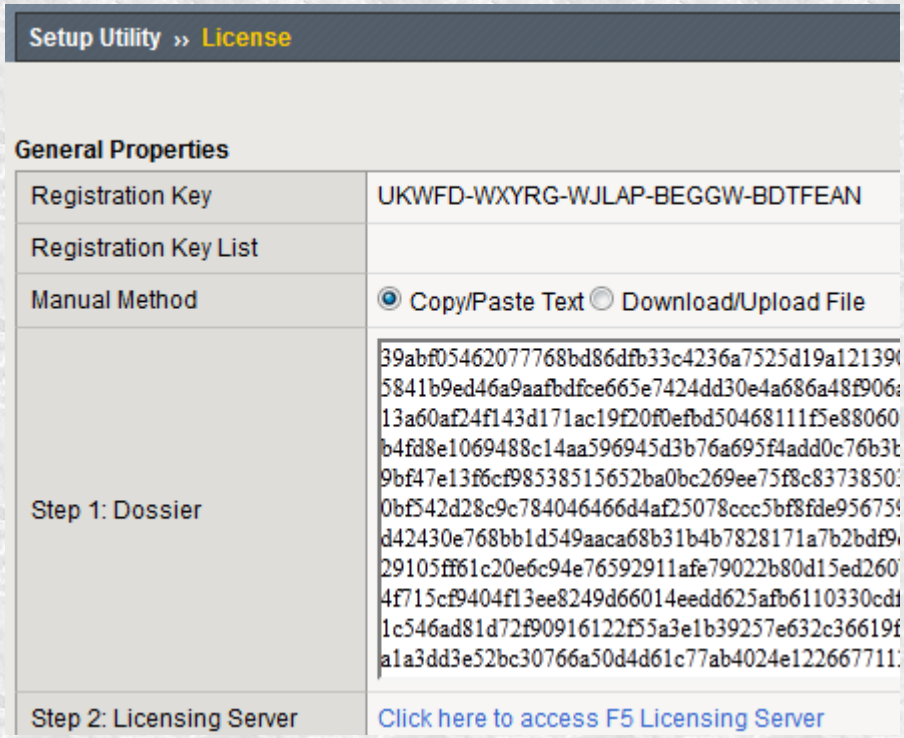
Mink van?


```
[root@localhost:NO LICENSE] config # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:66:FC:4B
          inet addr:192.168.233.128  Bcast:192.168.233.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe66:fc4b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:50 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5921 (5.7 KiB)  TX bytes:1836 (1.7 KiB)

eth1      Link encap:Ethernet  HWaddr 00:0C:29:66:FC:55
          inet6 addr: fe80::20c:29ff:fe66:fc55/64 Scope:Link
          UP BROADCAST RUNNING PROMISC ALLMULTI MULTICAST  MTU:1500  Metric:1
```

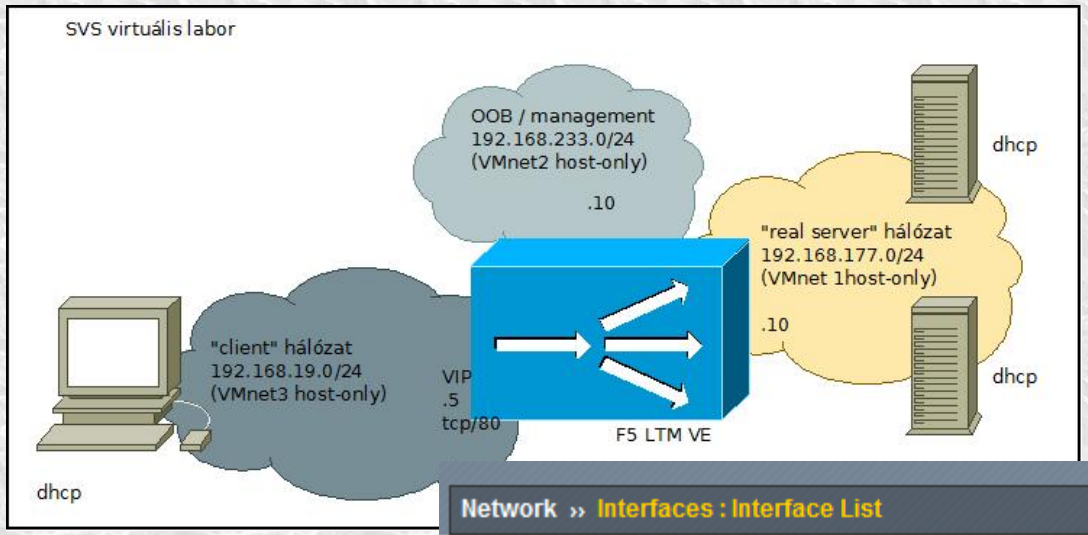
Licensing

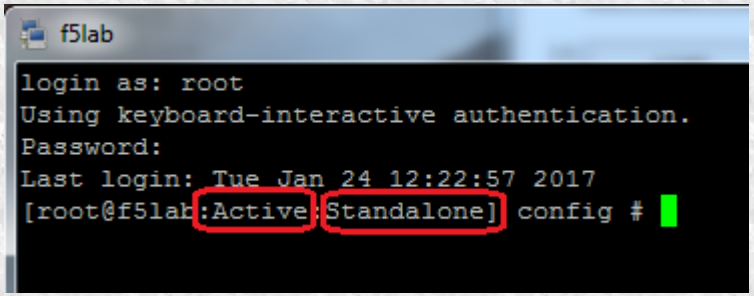
- Manuális aktiválást végzünk, de lehetne automatikusan
- Szükség van egy aktivációs kulcsra (lehet labor, éles vagy teszt kulcs is)
- Erre kapunk egy karaktersorozatot (dossier)
- Ezt az F5 licensz oldalon lehet becserélni egy licenszre (https://secure.f5.com/Infopage/)

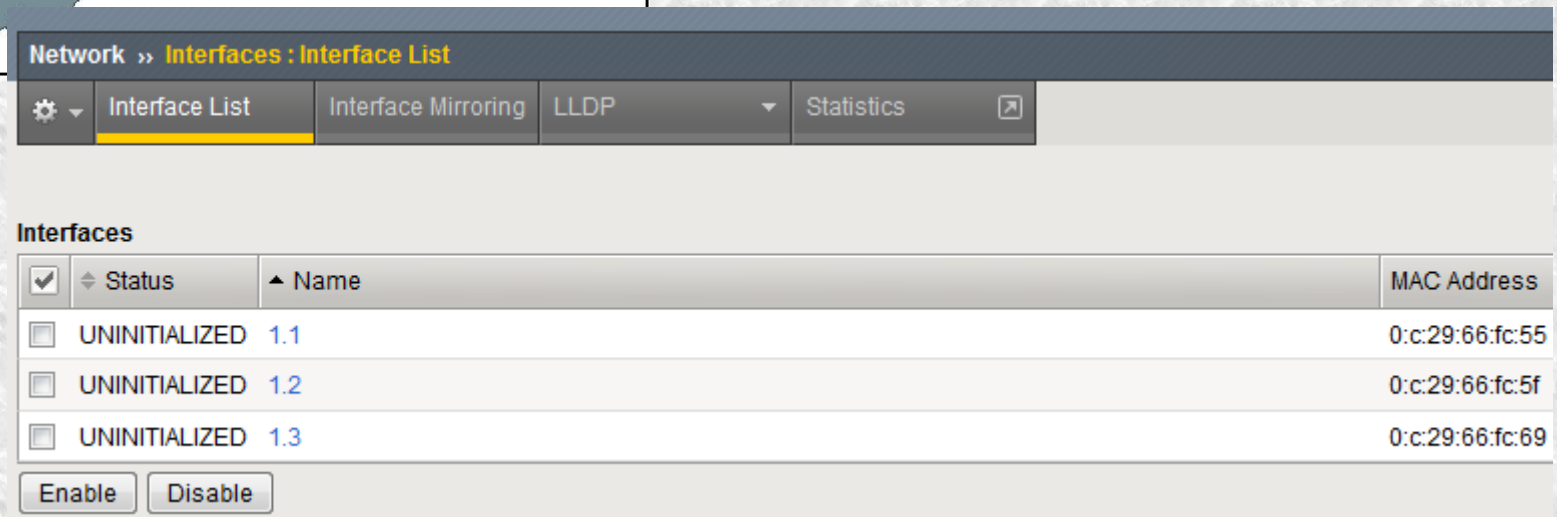


Első lépések

- Legfontosabb a hálózati interfészek beállítása úgy, ahogy használni szeretnénk

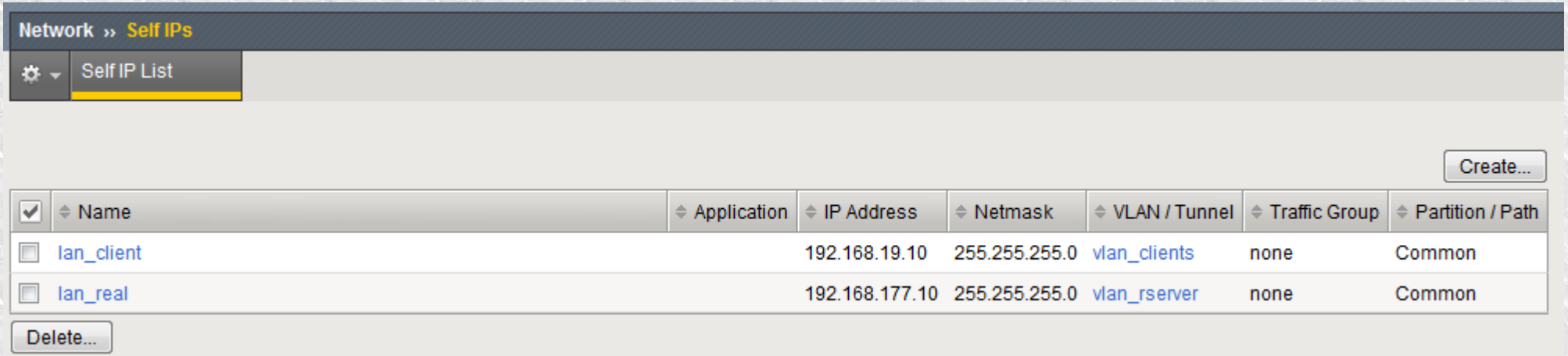






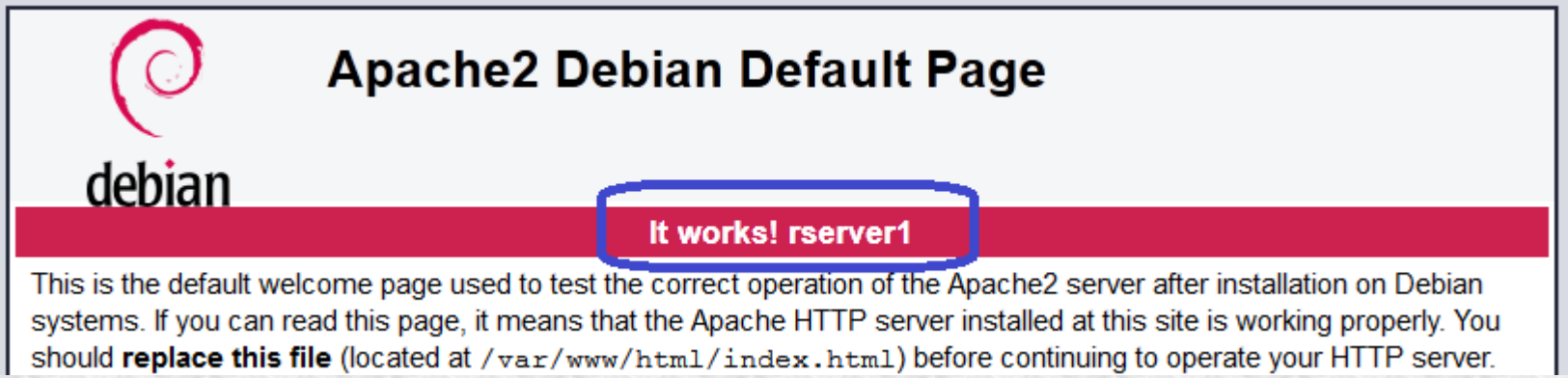
Első lépések

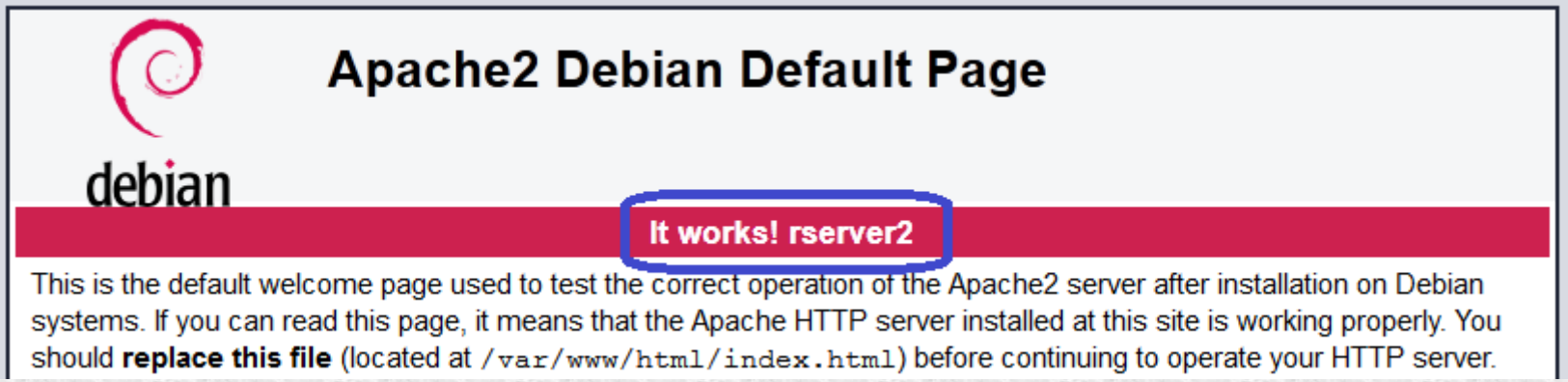
- Az egyszerűség kedvéért két interfész elegendő: egyik néz a valódi szerverek felé, a másik pedig a látogatók felé, ahonnan a kéréseket indítani fogjuk
- A szerverek és a klienseik közé L3 üzemmódban bekerült az LTM, a két hálózatot egymástól teljesen el kell választani, ehhez két külön VLAN-t használunk, amik azonban “untagged” módban vannak konfigurálva. Ez azért lényeges, mert a vmware, amiben az egész fut, nem számít VLANokra, így hiába küldene az LTM 802.1q kereteket.
- Emlékezz a korábbi oktatásokra, a VLAN fejlécbe szúrására és arra, mi történik, ha olyan végpont kapja a taggelt kereteket, ami nem számít erre!



A két rserver

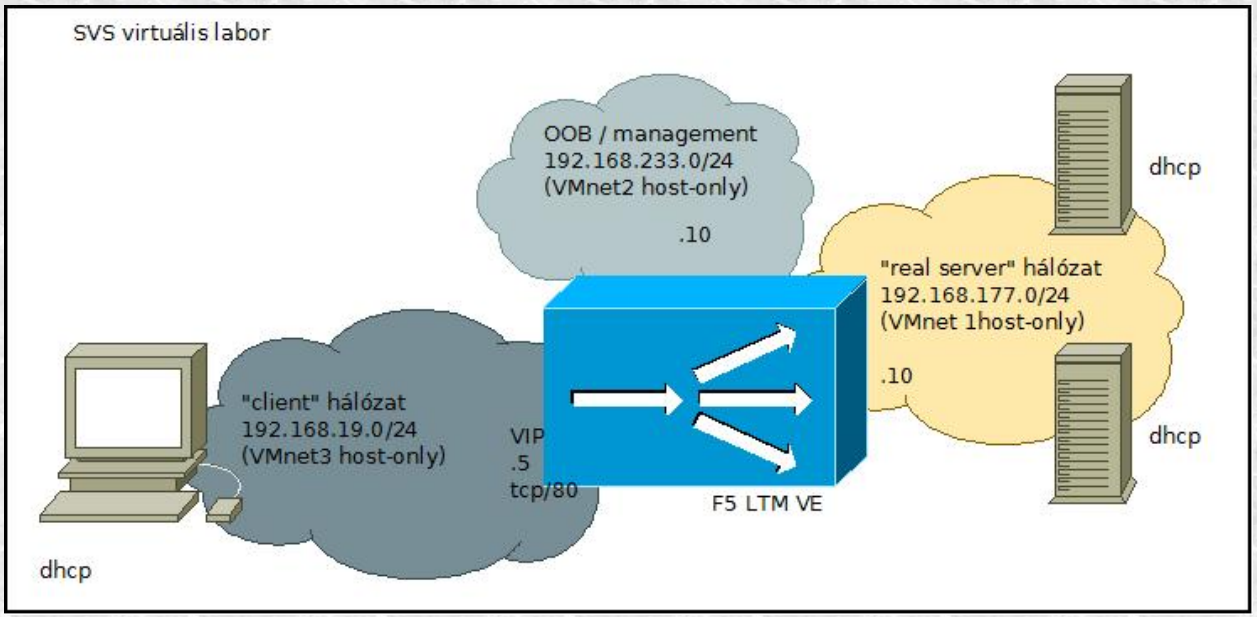
- A laborban két default install debian 8 van, amelyek az alapértelmezett “It works!” weboldalt jelenítik meg a látogatóknak. Mindkettőben ezt az oldalt kicsit megszerkesztjük, hogy egyedi legyen: hozzáadjuk a szerver nevét, hogy később tudjuk: melyik oldalt melyik szerver szolgáltatta (mert amúgy egyformák)





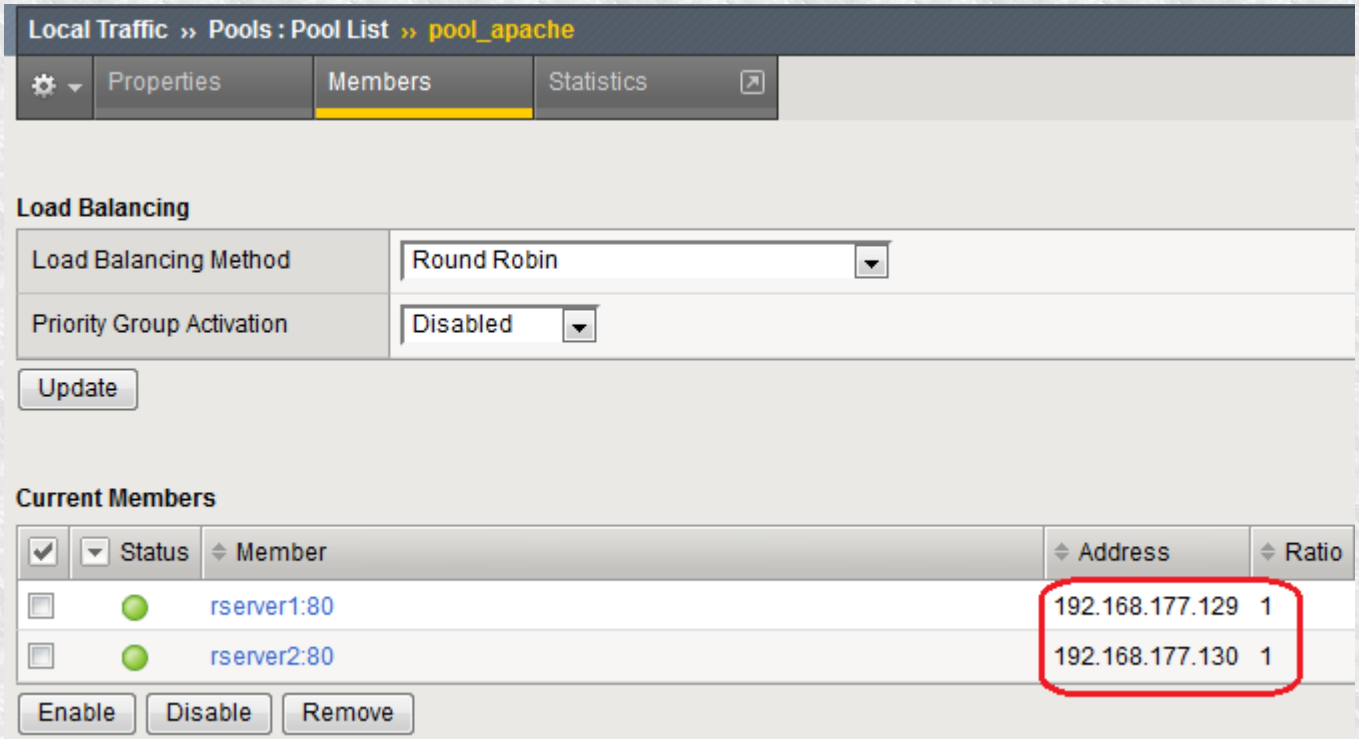
A VIP

- Ebben az állapotban a windows kliens csak azért éri el a két valódi szervert, mert a vmware, amiben futnak ugyanezen a gépen van. Kívülről ezek a gépek nem elérhetőek, mert:
  - Senki számára nem ismert, hogy a 192.168.177.0/24 hálózat merre van, hogyan kell oda eljutni
  - A valódi szervereknek csak a DHCP miatt van átjárójuk, amúgy nem tudnának kijutni az alhálózatukból
  - Itt, most, mivel a vmware host az átjáró, aki nem enged ki forgalmat, tulajdonképp nincs is átjáró
- A VIP egy másik hálózatban kell legyen, hiszen a kliensek ezt az IP címet hiszik “a szervernek”, ezt érik el
- Igazolandó a fenti állítást, egy virtualizált windows ha pingel a 192.168.19.0/24 -ből, nem lát át a szerverekig
- A valós élethelyzetet a virtuális windows mutatja, aki szimbolizálja az internet felől érkező látogatót



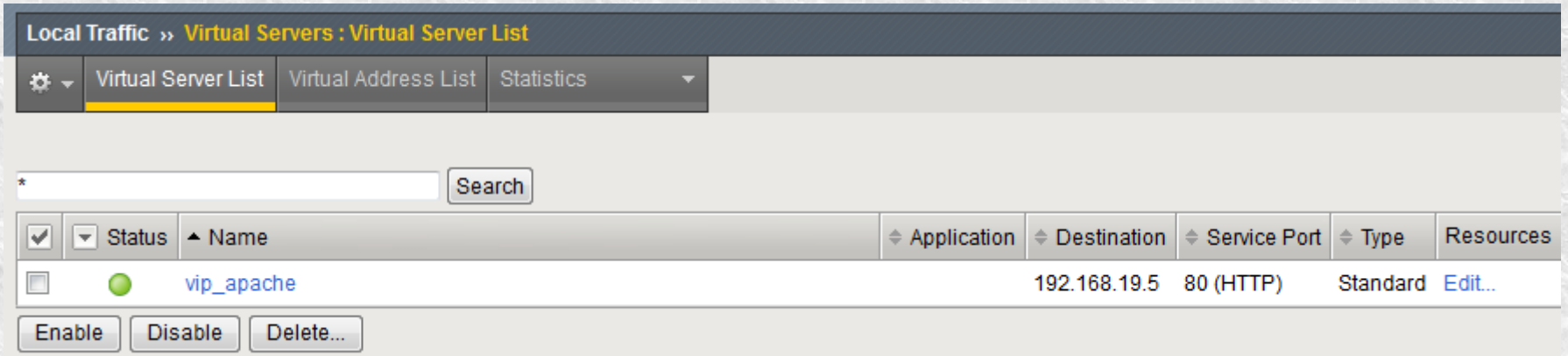
A VIP (2)

- Elsőként a szerverfarmot kell létrehozni, ami az azonos funkciójú valódi szerverek csoportja, az F5 világban Pool-ként ismert.
- A Pool tulajdonsága, hogy milyen arányban osztoznak a szerverek a forgalmon, hogyan történik a szerverek monitorozása, valamint egyéb paraméterek



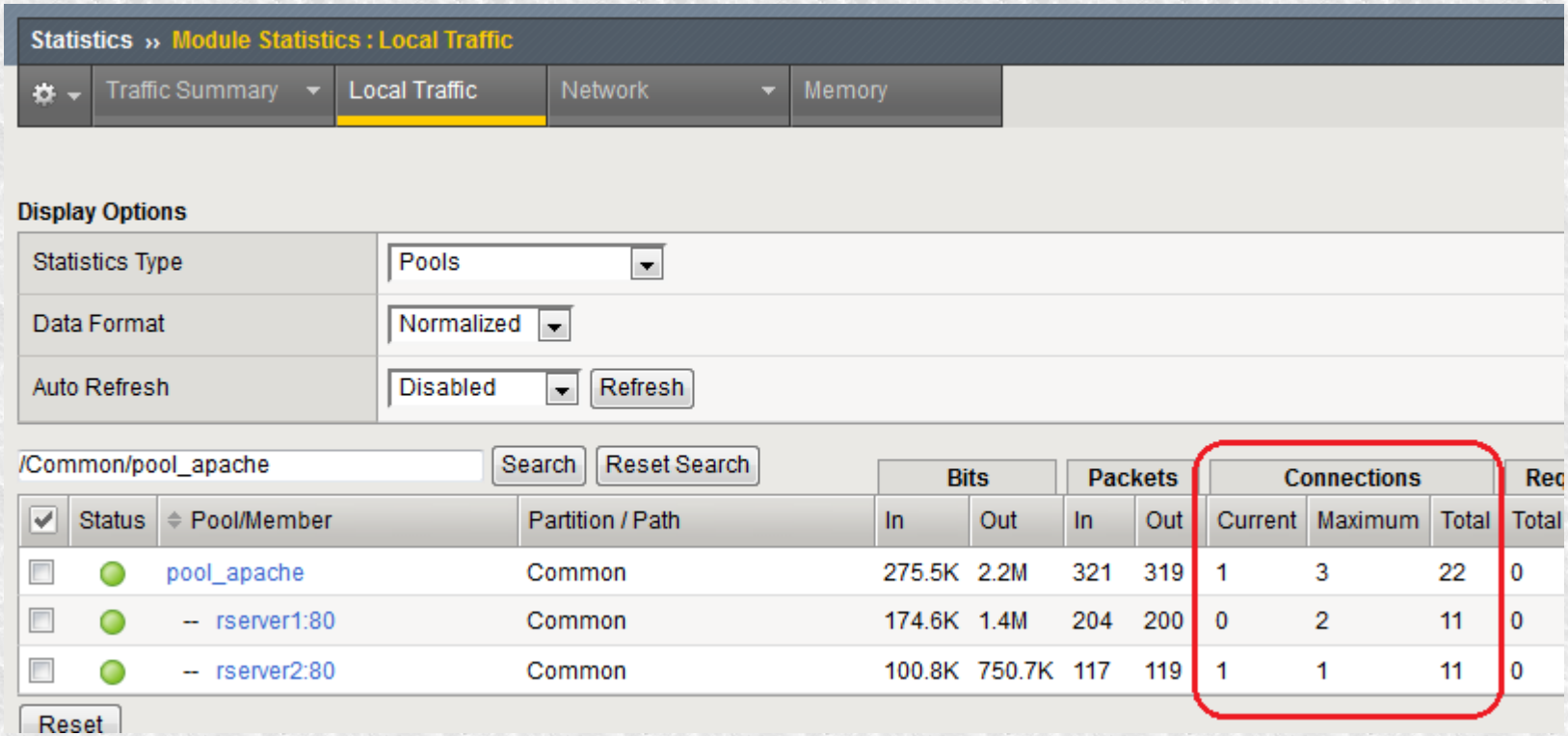
A VIP (3)

- source: honnan szolgálunk ki kéréseket? Nyilván bárhonnan
- destination: mi lesz a VIP konkréten és milyen porton fogadunk kéréseket? 80-as porton, de lehetne más
- vlan and tunnel traffic: melyik interfészen lesz elérhető a VIP? a saját vlanjában
- Source NAT: az előadás elején említettek szerint kell SNAT, legyen most auto map
- default pool: mely szerverek kapják meg a forgalmat? Egyetlen egy van csak, amit az előbb hoztunk létre



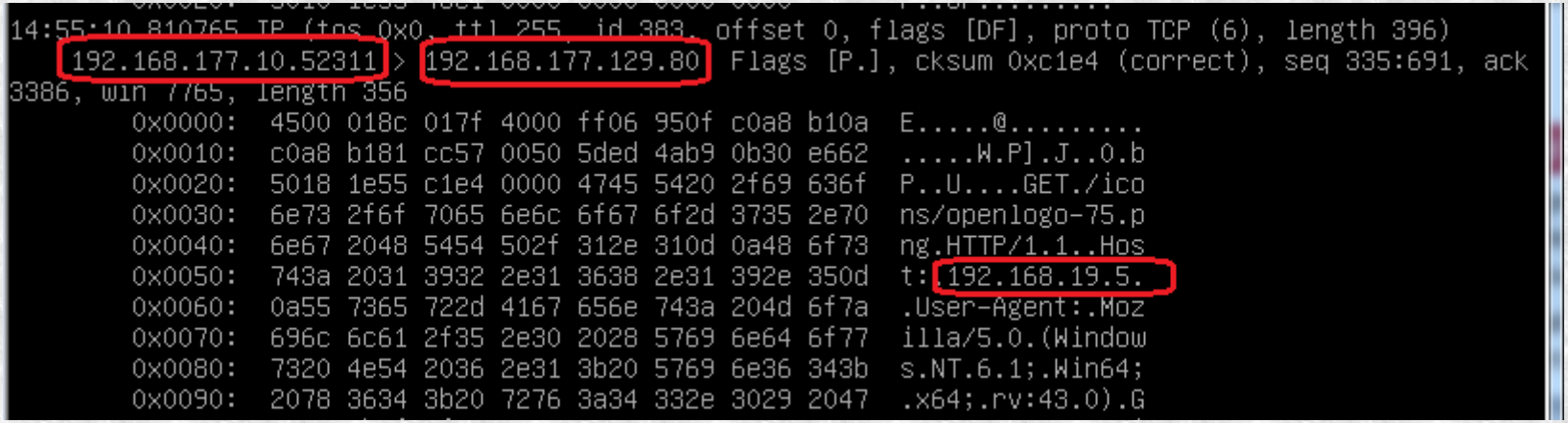
Teszt!

- Fontos felfedezés: ismét ellenünk dolgozik valami – a böngésző cache



SNAT

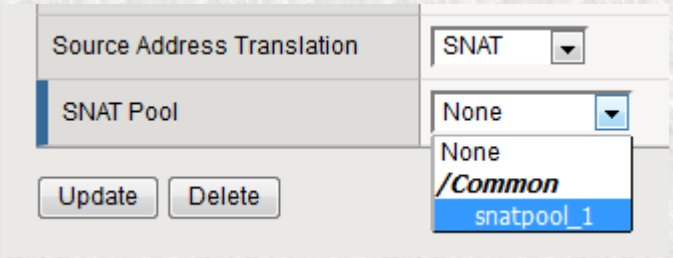
- Láthatjuk, hogy a valódi szerverek szerint minden kérés ugyanarról a címről érkezik
- Ez a SNAT következménye, pontosabban a SNAT auto map beállításnak
- A cím ismerős: ez az F5 saját címe (self-ip), tehát az F5 saját címét használja, hogy kiküldje a forgalmat a valódi szervernek és az vissza is találjon az eredeti feladóhoz
- Mint minden új technológia: ez is bevezet nehézségeket
- A lenti tcpdump a valódi szerveren készült, a beérkező forgalomról. Látható, hogy nem csak a forrás IP címmel van probléma, de a címzett IP cím (192.168.177.129) a hálózati rétegben sem egyezik azzal, mint ami az alkalmazási rétegben szerepel, amit a kliens küldött a HTTP fejlécben (192.168.19.5)
- Ez a valódi szerver üzemeltetőnek jelent(het) problémát, hogy olyan HTTP host fejléc tartalmakat fog látni, amelyek első ránézésre nem világos, hogy mit keresnek nála



SNAT (2)

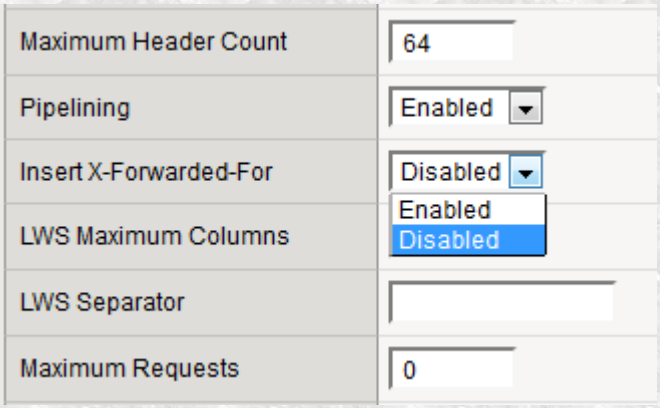
- A forrás IP cím ismerős: ez az F5 saját címe (self-ip), tehát az F5 saját címét használja az auto pool beállításnál, hogy kiküldje a forgalmat a valódi szervernek és az vissza is találjon az eredeti feladóhoz
- Ez többek között azt jelenti, hogy a szerver üzemeltetői nem ismerik a látogató eredeti IP címét
- Valamint ez azt is jelenti, hogy limitáltuk magunkat: egyszerre legfeljebb 64K kapcsolatot tudunk kezelni, ami bizonyos esetekben kevés lehet
- A HTTP kapcsolatállapot nélküli protokoll, a tartalom minden eleméhez a kliens új kapcsolatot nyit, tehát ha van 98 kép egy weboldalba ágyazva, az összesen 100 kapcsolat (98 kép, favicon, plusz az oldal maga)
- Tíz szerver, egyenként száz kliens, egyenként 100 kapcsolat: százezer kapcsolat, ami több, mint 64K!

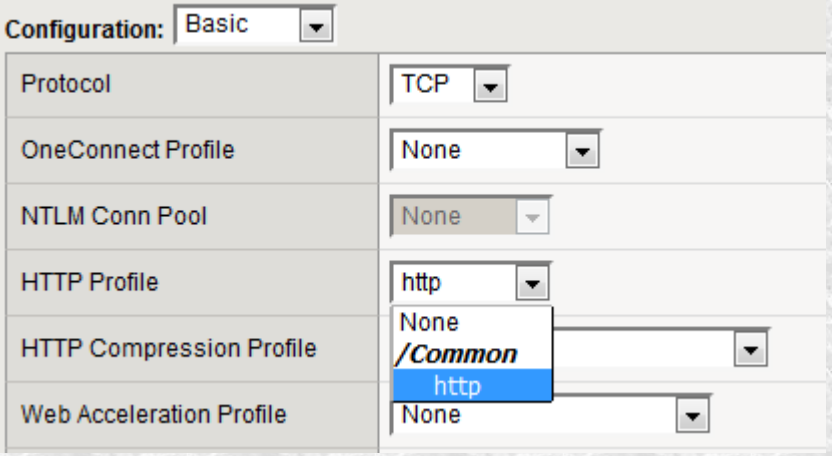
```
root@rserver-1:~# tail /var/log/apache2/access.log
192.168.177.10 - - [24/Jan/2017:15:38:06 -0500] "GET /i
://192.168.19.5/" "Mozilla/4.0 (compatible; MSIE 5.01;
192.168.177.10 - - [24/Jan/2017:15:38:07 -0500] "GET /i
://192.168.19.5/" "Mozilla/4.0 (compatible; MSIE 5.01;
192.168.177.10 - - [24/Jan/2017:15:38:08 -0500] "GET /i
://192.168.19.5/" "Mozilla/4.0 (compatible; MSIE 5.01;
192.168.177.10 - - [24/Jan/2017:15:38:09 -0500] "GET /i
://192.168.19.5/" "Mozilla/4.0 (compatible; MSIE 5.01;
192.168.177.10 - - [24/Jan/2017:15:38:10 -0500] "GET /i
://192.168.19.5/" "Mozilla/4.0 (compatible; MSIE 5.01;
192.168.177.10 - - [24/Jan/2017:15:38:11 -0500] "GET /i
://192.168.19.5/" "Mozilla/4.0 (compatible; MSIE 5.01;
192.168.177.10 - - [24/Jan/2017:15:38:12 -0500] "GET /i
://192.168.19.5/" "Mozilla/4.0 (compatible; MSIE 5.01;
```

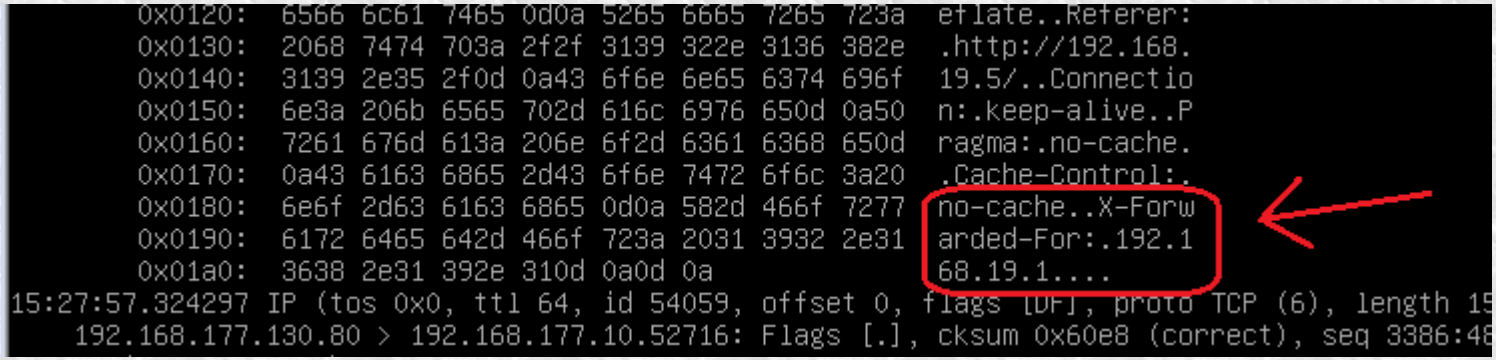


SNAT (3)

- Mi a teendő ha a valódi címek fontosak az alkalmazás üzemeltetőnek? HTTP fejléc, vagy coookie
- A fejlécbe a gyári HTTP profil módosításával, vagy iRule-lal lehet írni, cookie-t beszúrni is iRule tud
- A módosított HTTP profilt pedig használni is kell (eddig csak TCP profilt használtunk)

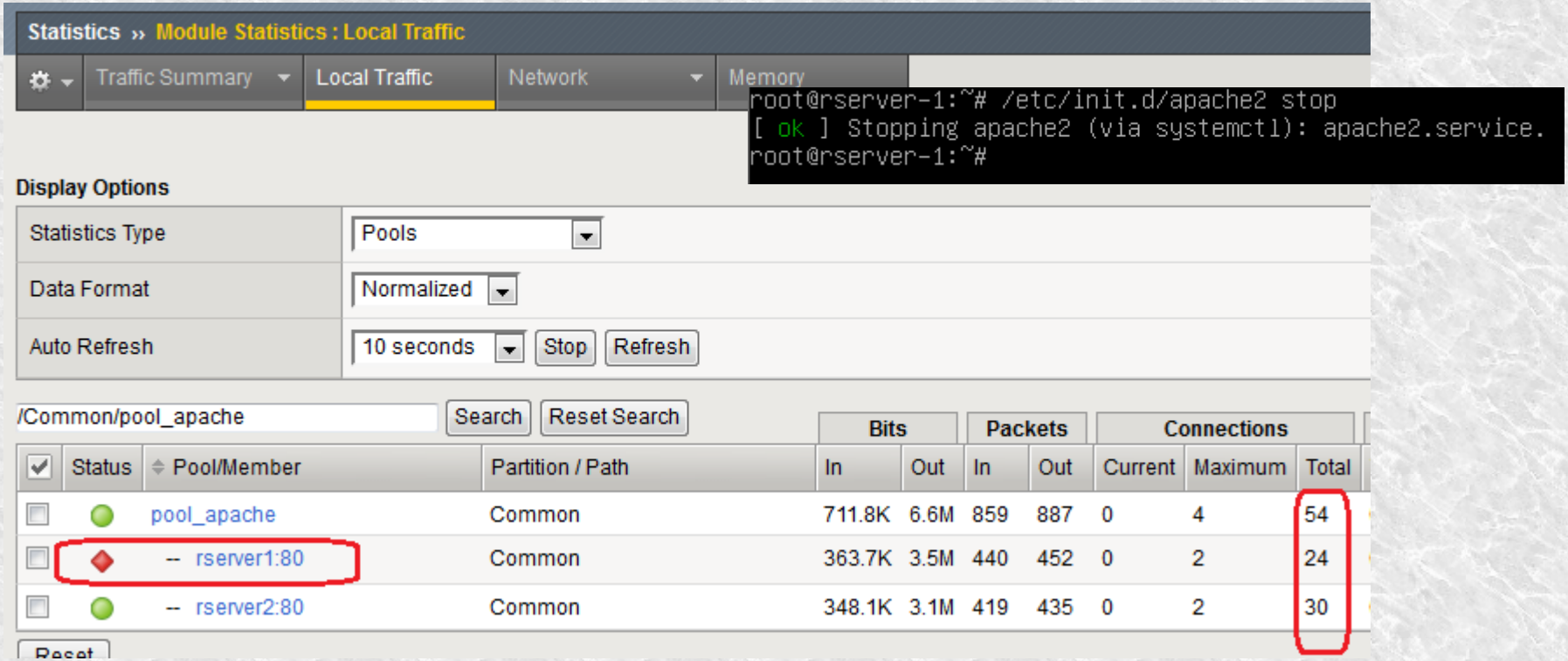






Hibák, hibák!

- Az egész előadás lényege a hibatűrő környezet felépítése volt. Mi történik hiba esetén?
- Mi történik, ha leállítok egy szervert? Mi történik, ha csak a szolgáltatást rajta?
- Itt lépnek be a képbe a monitorok, amik ráadásul örökölhetőek felettes objektumtól

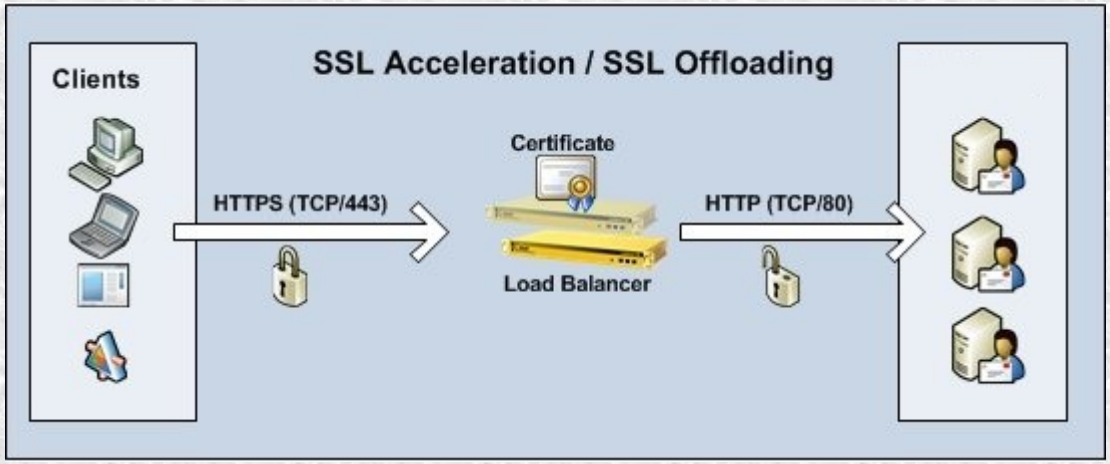


SSL offloading

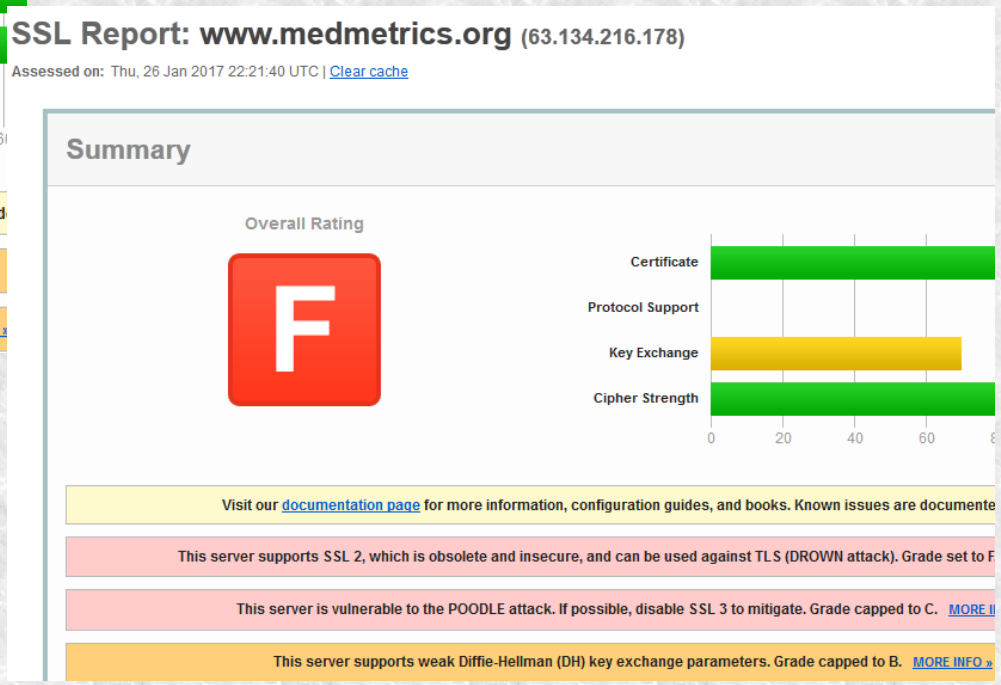
- Az erőforrás-igényes SSL feladatok elvétele a szerverektől és annak kezelése hardvertámogatás mellett, a terhelés-elosztó eszközön
- Egy F5 blade képes kezelni 2M/sec L7 kérést, 48M egyidejű kapcsolatot, 80 Gbit/s átvitel, 44k SSL TPS-t
- Egy 2400 viprion keretben négy ilyen blade lehet, egyenként 4 x 40 gbit ethernet csatlakozóval
- Erőforrás tehát van, jut feldolgozni a nagyszámú kapcsolatot

Előnyök

- **csak 2016-ban 34** sebezhetősége derült ki az openssl-nek, a legfontosabbak: heartbleed, poodle
- Mi rövidebb: ellenőrizni, patchelni, javítani, frissíteni 50 szervert, vagy két LB-t? És ha 34-szer kell ugyanezt?
- Persistence: cookie esetén, SSL felett ha nincs SSL offloading, az LB nem tud belenyúlni a fejlécbe!



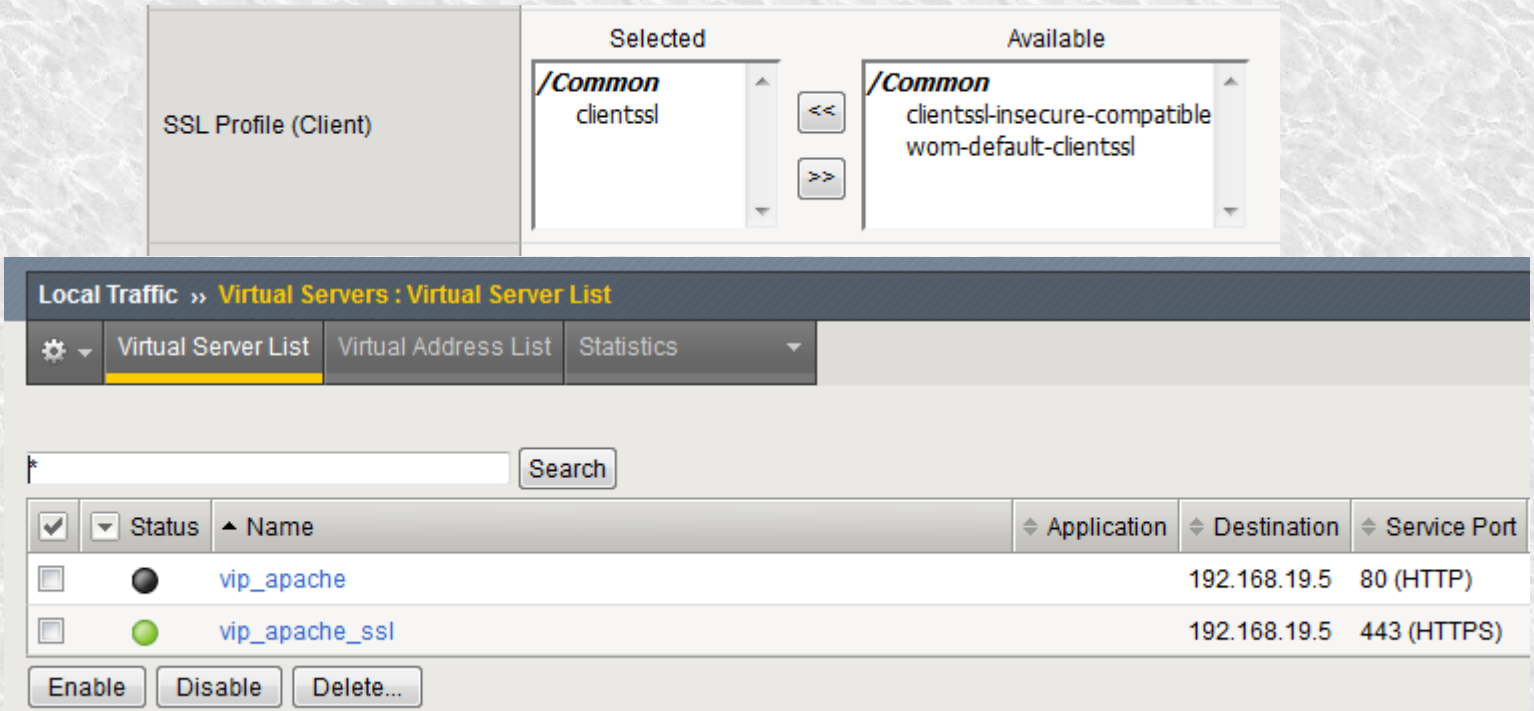
SSL offloading (2)



SSL offloading (3)

- Opcionális: ha nem tetszik a default tanúsítvány, első lépés: egy CA legenerálása
- Ha korrekt tanúsítványt szeretnél: második lépés: egy tanúsítvány kérelem létrehozása, aláírása
- A LB beállítása
- Tesztelés

```
$ openssl req -new -x509 -extensions v3_ca -keyout ca.key -out ca.crt -days 10
$ openssl req -new -nodes -out server.csr -keyout server.key
$ openssl ca -out server.crt -infiles server.csr
```



Örülök, hogy eljöttél meghallgatni. Kérdések?

Rólunk: **http://svs.cx**
Cikkek, útmutatók, leírások: **Tech Corner**

Kérlek, ne felejtsd el kitölteni a három kérdéses kérdőívet a mai estéről.

Köszönjük az eddigi visszajelzéseket. Ha nem tetszik, ahogy csináljuk, kérlek, mondd el nekünk. Ha tetszik, mondd el másoknak!

Korábbi oktatások anyagai elérhetőek Facebook oldalunkon át. Jövőbeli oktatások, kedvezményeket biztosító kódok valamint egyéb információk elérhetőek ugyanott, vagy a hírlevelünkben.

Feliratkozás: **http://svs.cx/lists/halozat-kezdo**