# Stateful load-balancing in swarm

<< Back to Docker main

## Contents



# Why an other load-balancer?

Azt elöljáróban elmondhatjuk, hogy a swarm-ot illetve, úgy általában a konténer cluster-eket els?sorban stateless micorservice-ekre találták ki, ahol egymás mellett akár több ezer stateless service példány fut, és tényleg teljesen mindegy hogy éppen melyik kapja a kérést. Ilyen méretekben nincs is értelme stateful szolgáltatásokról beszélni. A swarm mode is erre készül, a nagy számú, stateless végpontok kezelésére tökéletes a swarm-ba beépített Layer 4 (szállítási rétegbeli) load-balancer, ami Round-rubin módon mindig a következ? elérhet? konténernek adja a feladatot.

Azonban igen is szükség van stateful szolgáltatásokra, és bár akkora cluster nem építhet? bel?lük mint a stateless szolgáltatásokból, kiválóan passzolnak a swarm világba.

A konténeres cluster világban Session kezelést kétféle módon hozhatunk létre ha nagyon messzir?l nézzük:

- **Elosztott session kezelés**: minden egyes konténer, ami részt vesz a szolgáltatásban hozzáfér egy központi sesison tárhoz, ahonnan minden egyes hozzá beérkez? request estén ki tudja olvasni, hogy hol tart a session és onnan folytatja. El?nye, hogy ebben az esetben használni tudjuk a beépített layer 4 load-balancer-t, és nincs külön szükségünk swarm konfiguráció követésre, bármikor hozzáadhatunk új konténereket a cluster-hez ha szükség van rá. Azonban a szakirodalom szerint ez a megoldás nagyon rosszul skálázódik. Míg pár láb esetén remekül m?ködik, akár már 50 láb esetén is nagyon nehézkessé válik, hogy minden konténer tényleg megkapja megfelel? sebességgel a legfrissebb session-t. Nagy lábszám esetén ez gyakorlatilag kivitelezhetetlen.
- **Központosított session kezelés a load-balacer-ben**: Ebben az esetben a beépített Layer 4 load-balancer értelem szer?en nem használható, ezért itt egy olyan, küls? load-balancer megoldásra van szükség, ami tud sticky session-t kezelni, ezen felül képes automatikusan lekövetni a swarm cluster változásait és remekül bírja a terhelést.

A session kezelésen felül, még több okunk is lehet, hogy miért tegyünk egy load-balancing szolgáltatást a swarm cluster elé:

- SSL/TLS termination
- Content?based routing (based, for example, on the URL or a header)
- Access control and authorization
- Rewrites and redirects

Itt jön a képbe a Traefik ami azt állítja magáról, hogy erre találták ki.

# Statefull Load balancing with Traefik

https://boxboat.com/2017/08/03/deploy-web-app-docker-swarm-sticky-sessions/
http://www.littlebigextra.com/how-to-maintain-session-persistence-sticky-session-in-docker-swarm-with-multiple-containers/

## Áttekintés

A Traefik egy univerzális Layer 7 (http) load-balancer és reverse proxy. Direkt microservice környezetre találták ki és támogatja is gyakorlatilag az összes konténer orchestration platformot és többféle service discovery szolgáltatást is:

- Docker
- **Swarm mode** <<<
- Kubernetes
- Marathon
- Consul, Etcd --> service discovery
- Rancher
- Amazon ECS)

**A docker-swarm-ot natívan támogatja. Telepíthet? docker image-ként, és van hozzá webes információs felület is.**

**Warning**
A Traefik nem webserver, csak egy reverse proxy. Nem tud pl static tartalmat kiszolgálni!

A Traefik úgy m?ködik, hogy valamelyik swarm manager-tr?l periodikusan lekérdezi az aktuális swarm konfigurációt (szolgáltatások, és a szolgáltatáshoz tartozó konténerek). Ez alapján teljesen automatikusan konfigurálja magát és változás estén újra konfigurálja magát (pl. ha n? vagy csökken a cluster, vagy ha új szolgáltatás kerül telepítésre). Mivel magától leköveti a swarm cluster változásait, ideális megoldás a swarm-hoz mint Layer 7 load balancer.

A Traefik-et futtató VM publikus IP címén lesz elérhet? az összes Traefik által kezelt szolgáltatás. Íme egy példa egy lehetséges konfigurációra, hogy hogyan érhetünk el egy swarm szolgáltatást:

```
http://<traeif node public IP>/<PathPrefix>
```

A **PathPrefix**-et a swarm szolgáltatás telepítése közben kell megadni label-ek segítségével, tehát minden PathPrefix szolgáltatás specifikus. A Traefik nem kezd el automatikusan minden a cluster-re telepített szolgáltatáshoz load balancer/reverse proxy szolgáltatást nyújtani. A neki szánt szolgáltatásokat a szolgáltatás telepítése közben megadott traefik specifikus címék segítségével azonosítja be és konfigurálja.

Több fórumon is azt írják, hogy a Traefik-et csak valamelyik manager node-on lehet futtatni. Egyrészr?l ez nem igaz, másrészr?l hiba lenne ha így lenne. A manager node-ot egyrészt nem szabad load-balanc feladatokkal terhelni. Ha a manager-t túlterhelnénk, leállhat a swarm cluster-ünk. Másrészr?l másféle hardver konfigurációra van szükség load-balanceoláshoz mint swarm manager futtatásához nem is beszélve a t?zfal szabályokról, security megfontolásokról. (A manager node-koat nyilván nem lehet elérhet?vé tenni a publikus interneten). Ugyan a Traefik dokumentációból ez implicit nem derül ki, de ett?l még lehet remote worker node-on futtatni a Traefik-et.

A Traefik a docker swarm API-n keresztül olvassa ki a swarm adatait (szolgáltatások, címék és konténerek). Ezt vagy valamelyik manager lokális socket-jét csatlakozva teheti meg, vagy a docker remote API-n keresztül, ami általában TLS felett fut (pláne produkciós környezetben). Nyilván a legegyszer?bb ha az egyik manager node-ra telepítjük föl, és ott mount-oljuk a manager docker engine lokális socket-jét:

```
/var/run/docker.sock:/var/run/docker.sock
```

Így s swarm információkat ki tudja olvasni a lokális docker démonból. Azonban ezt csak tesztelésre szabad így megcsinálni, ahogy erre több helyen is felhívják a figyelmet. Ha megnézzük a Traefik konfigurációs leírásának docker szekcióját, találunk benne egy ilyet:

```
# Can be a tcp or a unix socket endpoint.
endpoint =
```

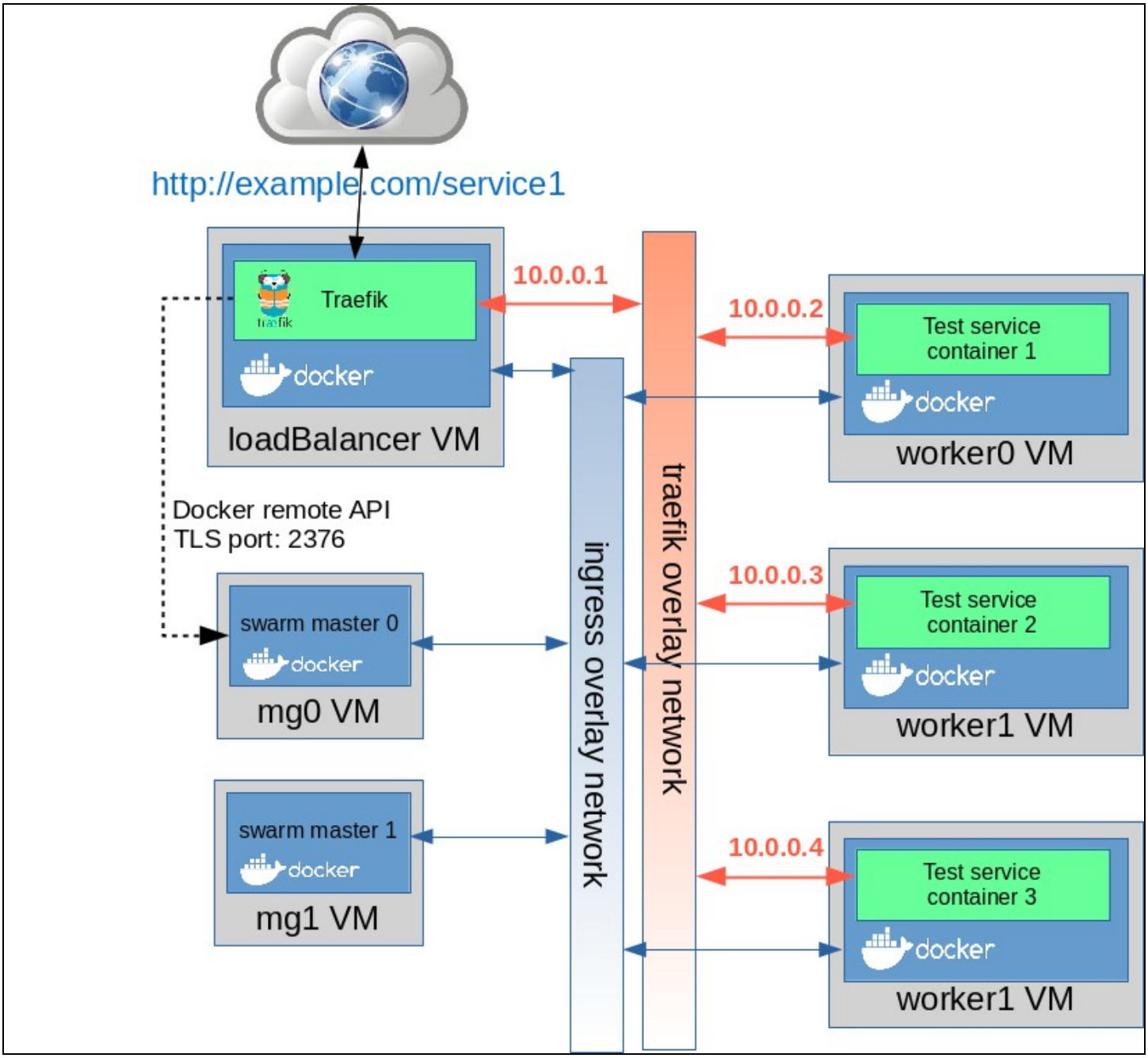
Ezen felül van benne egy TLS szekció is:

```
[docker.tls]
 ca = ...
```

Tehát képes távoli docker démonhoz kapcsolódni TLS felett a portainer-hez hasonlóan. Tehát ez a része kipipálva.

Van még egy fontos megkötés. A Traefik-nek közös overlay hálózaton kell lenni az összes olyan konténerrel, akiknek load-balancer szolgáltatást nyújt, mindjárt meglátjuk miért. Nyilván az ingress (routing mesh) hálózaton lév? konténer interfészek nem megfelel?ek layer 7 load balanc-olásra, mert ott már fut egy layer 4 load banacer, ami minden egyes kérésre másik node-ra irányítja a kérést (még akkor is ha direktbe a node IP címét adjuk meg), tehát a konténerek "publikus" IP címe nem megfelel?. Olyan konténer interfészekre van tehát szükségünk, aminek az IP címeit le lehet kérdezni a swarm master-t?l (szolgáltatásonként csoportosítva) de nem fut rajta a routing mesh, viszont fontos, hogy a load-balance-olás miatt a Traefik elérje ezen a hálózaton az összes konténert. Ezért létre kell hozzunk a Traefik számára egy új overlay hálózatot, amire egyrészt a Traefik, másrészt minden olyan konténer csatlakozik, ami olyan szolgáltatás része, amihez a Traefik load balancer szolgáltatást nyújt. Innent?l kezdve nem is számít, hogy a konténerek milyen portokat osztanak meg a host VM-el, mert mivel a Traefik közös hálózaton van a konténerrel, az abban futó összes alkalmazást eléri az alkalmazás által szolgáltatott porton. Tehát ha van egy szolgáltatásunk amit apache konténereket futtat a 80-as bels? porton, akkor attól függetlenül hogy a docker ezt a 80-as portot melyik host portra meppeli rá, a Traefik közvetlen meg tudja szólítani az apache-t a 80-as porton. Ebb?l kifolyólag még az sem számít ha több konténer is egy VM-en fut, ami ugyan ahhoz a szolgáltatáshoz tartozik, mivel a közös overlay network interfészek konténer szinten jönnek létre.

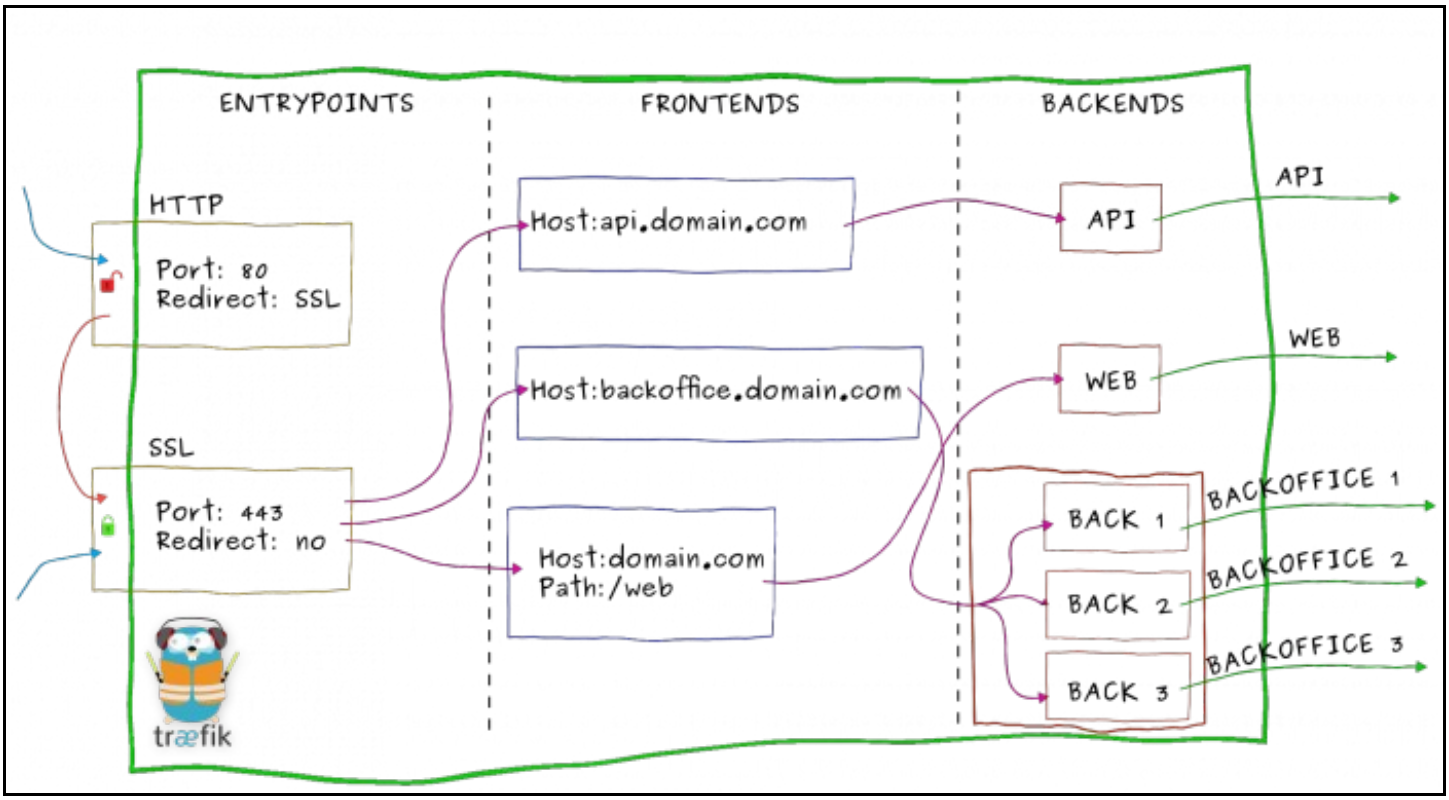
A Traefik-et futtathatjuk standalone módban is, docker nélkül egy távoli VM-en, viszont bárhol is fut, fontos, hogy rálásson a fent említett, közös overlay hálózatra. Nyilván ezt a legegyszer?bben úgy érhetjük el, hogy a Traefik-et is swarm szolgáltatásként indítjuk el a cluster-ben egy erre dedikált node-on, így nem kell külön azzal bajlódni, hogy egy cluster-en kívüli entitást csatlakoztassunk egy docker-es overlay hálózatra, ami nem lenne túl egyszer?. Így viszont a manager-ek ezt automatikusan megcsinálják.




A példában az összes kék doboz egy swarm cluster-be van kötve. A swarm clusre-nek két manager-e van, és három worker node-ja. A worker node-okra rá van telepítve egy darab swarm szolgáltatás, ami három példányban fut, miden node-on 1 konténer fut a test nev? szolgáltatásból. Szintén a cluster-re van telepítve a Traefik szolgáltatás, ami egy példányban fut, és ki van er?szakolva, hogy a loadbalancer nev? node-on hozza létre a swarm. A Traefik egy szem konténere a remote docker API-n keresztül rácsatlakozik a 0. számú manager node-ra, és onnan lekéri a szolgáltatások listáját. Meg fogja kapni a Teszt szolgáltatást. Ezután szintén a remote docker API-n keresztül le fogja kérni a Teszt szolgáltatást futtató node-ok IP cím listáját, ekkor fogja megkapni a következ? listát: 10.0.0.2, 10.0.0.3, 10.0.0.4. Innent?l kezdve, ha a http://example.com/test URL-re érkezik kérés, akkor mindig a 10.0.0.2,3, vagy 4-es IP -j? konténerre fogja irányítani a kérést a saját overlay hálózatán keresztül (az ábrán ezt hívják Traefik overlay hálózatnak). Látható, hogy ett?l függetlenül minden node csatlakozik a beépített ingress overlay hálózatra, de a Traefik load-balancing szempontjából annak most nincs jelent?sége.

**Warning**
Van egy kisebb probléma a Traefik jelenlegi architektúrájával, ami az ábrából is látszik. Jelenleg csak 1 darab remote docker API-t lehet konfigurálni a Traefik-nek, vagyis hiába van 3 manager node a cluster-ben, a Traefik sajnos csak egy dedikált manager node-hoz tud kapcsolódni, és ha az az egy manager node kiesik, akkor megsz?nik a load banacer is --> single point of failure (SPOF). Persze ez csak nagyon nagy cluster-eket érint? probléma. Ezt többen is feszegetik különböz? fórumokon, vannak rá különböz? hekmányolások, de szép megoldás még nincs rá

Anélkül, hogy nagyon mélyen belemennénk a Traefik m?ködésébe, annyit tudni kell róla, hogy három f? eleme van, amit az alábbi móricka ábrán is láthatunk:




- Entry point: ez kapja meg a kérést, vagy http-n vagy https-en. Leginkább arra jó, hogy a http-t átirányítsuk https-re. Ha a kérés túljutott az entry-point-on, megkapja a frontend szekció.
- A front-end szekció tartalmazza azokat a szabályokat, amik megmondják, hogy egy beérkezett kérést (a domain neve, a header mez?k vagy a path alapján) melyik beck-end-re kell továbbítani.
- A back-end tartalmazza a swarm service végpontok listáját (ezen a szinten már mindenképp service-r?l beszélünk és nem stack-r?l, mert még akkor is ha compose fájllal hoztuk létre a szolgáltatást, a Traefik service szinten van definiálva) Ha az entry-point-ról a kérés a frent-end szabályok segítségével eltalált a megfelel? back-end-re, akkor a Traefik továbbítja a kérést a kiválasztott tényleges docker konténernek.

# El?készületek

## Overlay hálózat definiálása

Ahogy azt már láthattuk, szükség van egy dedikált overlay hálózatra, amire a Traefik konténer és az összes többi olyan konténer is rá lesz csatlakozva, amik részt vesznek a load-balancing-ban.

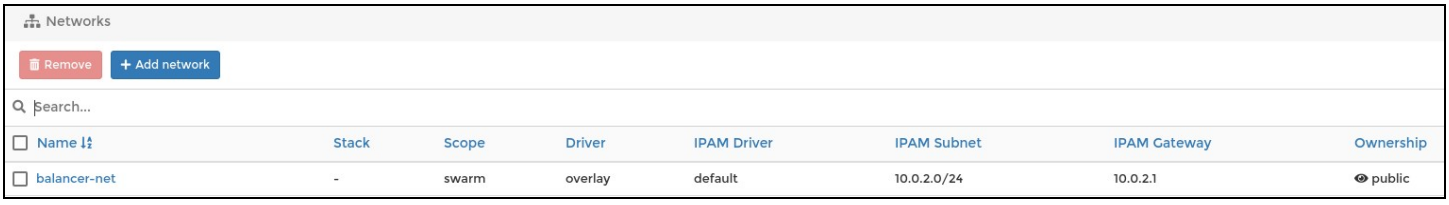
Bármelyik manager node-on futtassuk le az alábbi parancsot. Ha nem fontos az IP tartomány, akkor simán a **docker network create** paranccsal hozzuk létre az új hálózatot mindösszesen a **-d overlay** paraméter megadásával.

```
# docker-machine ssh mg0 docker network create -d overlay balancer-net
o4rhm35gkh24cd25rdt7hsm62
```

Nézzük meg (szintén a manager node-okon). Látható, hogy létrejött a swarm scope-ú új hálózatunk.

```
# docker-machine ssh mg0 docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
o4rhm35gkh24        balancer-net        overlay             swarm <<<<
e7b191c598c3        bridge              bridge              local
4648968db4af        docker_gwbridge     bridge              local
f1ab56710cf2        host                host                local
mzwld5ddadk6        ingress             overlay             swarm
b9b55fc2d01d        none                null                local
```

Megnézhetjük a Portainter-ben is az új hálózatot a Networks menüpontban:




## Load balancer node elkészítése

Els?ként készíteni fogunk egy új VM-et kifejezetten a load balancer számára, és ezt be fogjuk léptetni a swarm cluster-be. Létre fogunk hozni egy címkét is az új node-nak: **loadbalancer=true**. Ezzel fogjuk kikényszeríteni, hogy a Traefik swarm szolgáltatás egy szem konténere erre a node-ra települjön, ezen felül szintén ezzel a címkével fogjuk elérni, hogy semmilyen más konténert ne tegyen erre a node-ra a swarm.

Az alábbi script létrehozza az új VM-et, belépteti a cluster-be, és rárakja a címkét:

```
#!/bin/bash

#Get worker token
WORKER_TOKEN=`docker-machine ssh mg0 docker swarm join-token -q worker`

#Create load balancer node

docker-machine create -d kvm --kvm-network "docker-network" --kvm-disk-size "5000" --kvm-memory "800"  loadBalancer

docker-machine ssh loadbalancer docker swarm join --token $WORKER_TOKEN $(docker-machine ip mg0)

docker node update --label-add loadbalancer=true loadbalancer
```

Lépjünk be valamelyik manager node-ra és ott kérdezzük le az új **loadbalancer** nev? node címkéit. Látnunk kell hogy rendelkezik a **loadbanacer=true** címkével.

```
# docker node inspect --format='{{.Spec.Labels}}' loadbalancer
map[loadbalancer:true]
```

## Traefik telepítése

A Traefik-et docker service-ként telepíteni kell a cluster-re, úgy hogy garantáltan loadbalancer nev? node-ra kerüljön, valamint csatlakozzon a **balancer-net** nev? overlay hálózatra. Ezen a ponton több lehet?ségünk is van. Vagy a **docker service create** paranccsal definiáljuk az új szolgáltatást, vagy írunk egy **compose (yml)** fájlt, amiben a többi swarm szolgáltatással együtt telepítjük a Traefik-et is. Bármelyiket is választjuk, még azt is eldönthetjük, hogy megadunk a Traefik szolgáltatásnak (mivel csak egy példány lesz, mondhatnám azt is, hogy a Treaefik konténernek) egy küls? konfigurációs fájt, vagy cmd argumentumokkal adjuk meg a teljes konfigurációt.

**Note**
Ne feledjük el, hogy csak is azért telepítjük swarm szolgáltatásként a Traefik-et, hogy egy mozdulattal rá tudjuk kötni egy közös docker overlay hálózatra, amin azok a konténerek is lógnak majd, akik olyan szolgáltatáshoz tartoznak, akiknek load balnacer szolgáltatást kell nyújtani. Ez ahhoz kell, hogy a Traefik le tudjon kérdezni olyan végpont listát a manager node-tól, ami független a routing mesh-t?l, és amin keresztül a Traefik el is éri a szóban forgó konténereket.

### Konfiguráció elkészítése

https://docs.traefik.io/configuration/backends/docker/

A Traefik-nek van egy saját konfigurációs fájlja ami a Traefik konténerben lakik a **/etc/traefik/traefik.toml** helyen. Vagy ezt a fájlt felülírjuk vagy ezen fájl egyes értékeit írjuk felül CMD argumentumokkal a konténer definiálásakor. Azt szeretnénk, hogy a Traefik a swarm manager-rhez a remote TLS API-k keresztül csatlakozzon, ehhez meg kell adni a CA, a Cert és a titkos kulcsot fájlokat is a csatlakozáshoz. Ezen felül meg kell adjuk a manager node IP címét és TLS portját is. Mivel ennyi paramétert kéne megadni, célravezet?bb ha ezeket a Traefik konfigurációs fájljában adjuk meg

Ahhoz hogy a Traefik csatlakozni tudjon a manager node-hoz a docker remote API-n keresztül lényegében ugyan arra van szükség, amiket a Portainer remote kapcsolódásához beállítottunk a Monitorozás cím? fejezetben:

- manager node IP címe + secure remote port (192.168.42.75:2376). Emlékezzünk rá, hogy a 2376 secure port a boot2docker oprendszerben defualt nyitva van.
- TLS CA cert: ca.pem
- TLS certificate: cert.pem
- Secret key: key.pem

Az utóbbi hármat a **docker-machine** már legyártotta a VM létrehozásakor, hogy jelszó nélkül, a **docker-machine ssh** paranccsal csatlakozni tudjunk a VM-hez. Ugyan ezekre van itt is szükség. A publikus kulcsunkat pedig a VM létrehozásakor már a helyére másolta. (azt hiszem a ca.pem-el tudjuk ellen?rizni a VM tanúsítványát, a VM a mi cert.pem-ünkel ellenörzi a mi kilétünket, és a titkos kulccsal hozzuk létre az ssh kapcsolatot. (ssh -i key.pem loadbalancer)

A **/etc/traefik/traefik.toml** fájlban minimum az IP címét a portot és a három TLS fájl helyét kell hogy beírjuk. Ehhez létre fogunk hozni egy új **traefik.toml** fájlt a Traefik-et futtató VM-en, és azt fel fogjuk csatolni bind mount-al az egy szem Traefik konténerben az **/etc/traefik/traefik.toml** helyre, így el fogjuk fedni a konténerben lév? fájlt.
Induljunk ki ebb?l: https://github.com/containous/traefik/blob/master/traefik.sample.toml
És írjuk felül az alábbiakat:

```
################################################################
# Docker Swarm Mode Provider
################################################################

# Enable Docker Provider.
[docker]

# Docker server endpoint.
# Can be a tcp or a unix socket endpoint.
endpoint = "tcp://192.168.42.75:2376"

# Default domain used.
# Can be overridden by setting the "traefik.domain" label on a services.
#
# Optional
# Default: ""
#
domain = "docker.localhost"

# Enable watch docker changes.
#
# Optional
```

```
# Default: true
#
watch = true

# Use Docker Swarm Mode as data provider.
#
# Optional
# Default: false
#
swarmMode = true


# Expose services by default in Traefik.
#
# Optional
# Default: true
#
exposedByDefault = true

# Enable docker TLS connection.
#
# Optional
#
[docker.tls]
  ca = "/etc/ssl/ca.pem"
  cert = "/etc/ssl/cert.pem"
  key = "/etc/ssl/key.pem"
  insecureSkipVerify = true
```

## Fájlok másolása

Mind a Traefik.toml fájlt, mind a TLS fájlokat (két cert + egy key) a loadbalancer VM-re kell másolni, hogy fel tudjuk ?ket mountolni a Traefik konténeren.

**Note**
Bárhol is futtatjuk a docker service create parancsot, az abban megadott **--mount** paraméter arról a VM-r?l próbálja meg mountolni a megadott mappát/fájlt, amire a swarm kiosztja a konténert. Mivel mi ki fogjuk er?szakolni label-ek segítségével, hogy a Traefik szolgáltatás egy darab konténere a loadbalancer nev? VM-en jöjjön létre, ezért fontos, hogy a fent említett fájlokat a VM-re másoljuk

A fájlokat a lokális gépr?l a **docker-machine** által létrehozott VM-kre a **docker-machine scp** paranccsal lehet átmásolni, aminek a szintaktikája szinte teljesen megegyezik a linux-os scp-vel.

Els?ként nézzük meg, hogy mi a home mappánk boot2docker oprendszerben ha ssh-val belépünk a loadbalancer VM-re. Látható, hogy ez a **/home/docker**, ide fogjuk másolni az TLS fájlokat és a Traefik konfigurációs fájlt is.

```
# docker-machine ssh loadbalancer pwd
/home/docker
```

A docker-machine scp parancs szintaktikája az alábbi:

```
docker machine scp /path/to/local/file MACHINE-NAME:/path/to/remote/file
```

Ezen felül a **-r** kapcsolóval rekurzívan másolhatunk mappákat.

Els?ként másoljuk a lokális **ssl** mappa tartalmát (amiben a három TLS fájl van) a **loadbalancer** nev? VM **/home/docker** mappájába. (mivel a kett?spont után nem adtam meg semmit, ezért a home mappába fog kerülni)

```
# docker-machine scp -r ssl loadbalancer:
key.pem           100% 1675     2.1MB/s   00:00
ca.pem            100% 1029     1.6MB/s   00:00
cert.pem          100% 1070     1.5MB/s   00:00

# docker-machine ssh loadbalancer ls -l ssl
total 12
-rwxr-xr-x    1 docker   staff         1029 Jul 28 13:24 ca.pem
-rwxr-xr-x    1 docker   staff         1070 Jul 28 13:24 cert.pem
-rwxr-xr-x    1 docker   staff         1675 Jul 28 13:24 key.pem
```

Most másoljuk a traefik.toml fájlt szintén a home mappába.

```
# docker-machine scp traefik.toml loadbalancer:
traefik.toml     100% 4585     7.1MB/s   00:00

# docker-machine ssh loadbalancer ls -l
total 12
drwxrwxr-x    2 docker   staff          100 Jul 28 13:24 ssl
-rw-r--r--    1 docker   staff         4585 Jul 28 13:26 traefik.toml
```

## Traefik szolgáltatás létrehozása

A Traefik-et a **docker service create** paranccsal fogjuk létrehozni. A teljes parancs az alábbi:

```
docker service create -d -p 8080:8080 -p 80:80 --name loadbalancer \
--mount type=bind,src=/home/docker/traefik.toml,dst=/etc/traefik/traefik.toml \
--mount type=bind,src=/home/docker/ssl,dst=/etc/ssl --constraint node.labels.loadbalancer==true \
--network balancer-net traefik
```

- **-p 8080:8080** - Ez a Traefik webes konzoljának a portja. Ezt a loadbalancer nev? VM 8080 portjára kötjük rá.
- **-p 80:80** - A 80-as porton fogja nyújtani a load-balancer szolgáltatást a Traefik (ha https is lenne, akkor a 443-at is meg kéne adni). Ezt szintén a loadbalancer VM 80-as portjára kötjük rá.
- **--mount type=bind,src=/home/docker/traefik.toml,dst=/etc/traefik/traefik.toml** - A loadbalancer VM-en lév? traefil.toml konfigurációst fájlt mountoljuk a Traefik konténer /etc/treafik/traefi.toml pontjára, amivel elfedjük a default fájlt, és így a konténer a mienket fogja látni.
- **--mount type=bind,src=/home/docker/ssl,dst=/etc/ssl** - mivel a konfigurációs fájlban a /etc/ssl-t adtuk meg, ugyan ide kell mountolni a TLS fájlokat a Traefik konténerben.
- **--constraint node.labels.loadbalancer==true** - Ezzel azt mondjuk meg, hogy kizárólag olyan node-ra telepíthet?, ami rendelkezik ezzel a címkével
- **--network balancer-net** - Rákötjük a Traefik service -t az újonnan létrehozott overlay hálózatra. Ide fogjuk rákötni azokat a service-eket vagy stack-eket is, kiket load balance-olni akarunk.

## Mi jött létre

Listázzuk a futó szolgáltatásokat. Látható, hogy a **loadbalancer** nev? szolgáltatás 1 példánnyal elindult

```
# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
04cj4vdsu0qu        loadbalancer        replicated          1/1                 traefik:latest      *:80->80/tcp, *:8080->8080/tcp
```

Listázzuk a loadbalancer nev? szolgáltatást. Látható, hogy a **loadbalancer** nev? node-ra tette, ahogy azt akartuk.

```
# docker service ps loadbalancer
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE             ERROR
v524v9oqqpz7        loadbalancer.1      traefik:latest      loadbalancer        Running             Preparing 9 seconds ago
4vnxiyz8xwoa         \_ loadbalancer.1  traefik:latest      loadbalancer        Shutdown            Rejected 9 seconds ago    "invalid mount
```

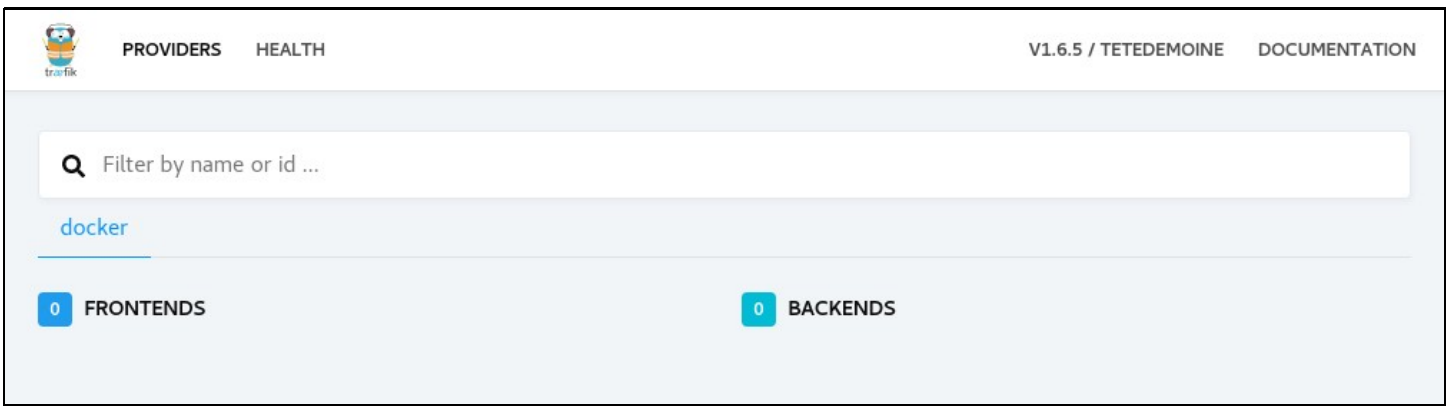
**Tip**
Ha valamiért nem tudna elindulni a Traefik szolgáltatás, (pl. mert hibásan adtuk meg a mountokat) akkor a swarm folyton meg fogja próbálni újra létrehozni a service-t. Az elhalt példányokat _ -fogja jelölni. Mivel máshogy nem adtuk meg, alapértelmezetten mindig újra indítja a swarm, ezért mindig keletkezik egy új task, (a régi mindig "Shutdown" állapotba kerül.

A Portainer-ben is meg kell jelenjen a service listában:


Most már el kell érjük a Traefik webes konzolt a loadbalancer VM publikus IP címén, a 8080 porton.

```
# docker-machine ip loadbalancer
192.168.42.42
```

Tehát itt: http://192.168.42.42:8080



Ahogy azt már láttuk, a Traeif-kben három lépcs?n jut el a kérés a docker konténerekig. Els?ként az Entry-point megkapja a kérést, majd a Front-end szabályok megmondják, hogy melyik back-end-re menjen tovább a kérés. A Back-end pedig swarm szolgáltatásokat szimbolizál, abban vannak azok a végpontok, amik közül választani fog egyet a Traefik mikor továbbítja a kérést.
A Traefik konzolt majd Load balancing test cím? fejezetben nézzük meg részletesebben, mikor már lesznek benne szolgáltatások.

## Swarm stack készítése

Most definiálni fogunk egy swarm stack-et, amihez a Traefik load-balancer szolgáltatást fog nyújtani. Ehhez els?ként el fogjuk készíteni a stack yml fájlját. (Használhatjuk a **docker service create** parancsot is, a lényeg, hogy megadjuk a Traefik-et vezérl? címkéket)

### Stack definiálása

Az új szolgáltatáshoz a **tutum/hello-world** image-et fogjuk használni, amiben fut egy apache, és a lending page-en (index.php) kiírja a konténer host nevét (konténer ID-t). Ez azért jó választás, mert remekül tesztelni lehet vele a stick-session kezelést, láthatjuk majd hogy ha egyszer már beestünk egy lábra, végig ott is maradunk, de session törlés után megint egy új lábat kapunk. Ezen felül az index.php-n még egy kép is található, tehát az url rewrite-ot is tesztelhetjük.

A szolgáltatás neve **helloworld** lesz. A **helloword.yml** fájl tartalma az alábbi:

```
version: "3"

services:

  helloworld:
    image: tutum/hello-world
    networks:
      - balancer-net
    ports:
      - "80"
    deploy:
      restart_policy:
        condition: any
      mode: replicated
      replicas: 5
      placement:
        constraints:
          - node.role == worker
          - node.labels.loadbalancer != true
      update_config:
        delay: 2s
      labels:
        - "traefik.docker.network=balancer-net"
        - "traefik.port=80"
        - "traefik.frontend.rule=PathPrefixStrip:/hello/"
        - "traefik.backend.loadbalancer.sticky=true"


networks:
  balancer-net:
    external: true
```

**Note**
Ha implicit nem mondjuk meg a compose -nak, hogy a hálózat már létezik, akkor a létre fog mindig hozni egy új hálózatot a **<service név>_<hálózat név>** néven. Tehát a fenti compose fájlból a **helloworld_balancer-net** hálózat jönne létre. Ha a hálózat mér létezik, akkor ezt az **external: true** paraméterrel jelezni kell.

Lehet?ség van rá, hogy más nevet használjunk a compose fájlban, mint a hálózat valódi neve. Ekkor az external után a name paraméterrel kell megadni a nevét:

```
external:
  name: balancer-net
```

Egy kis magyarázat a compose fájlhoz:
A teljes **deploy** szekció a **docker stack**-nek szól, a **docker compose** ezt a részt figyelmen kívül hagyja. Itt kell megadni a swarm specifikus beállításokat a szolgáltatáshoz.

- **networks:balancer-net**: Fontos, hogy a szolgáltatás összes konténere rá legyen kötve a közös overlay hálózatra, amire a Traefik is rá van kötve. Itt minden egyes konténernek egyedi IP címe van, még akkor is, ha egy node-on több konténer is létrejött. A Traefik ezen a közös hálózaton fogja megszólítani a konténereket, így nem lehet port ütközés (minden ip:port egyedi)
- **ports:"80":** Ezzel megmondtuk a docker-nek, hogy a konténer 80-as portját kösse össze a host VM egy véletlen választott portjával (mivel a küls? por itt nincs megadva). Szerintem ez a Traefik szempontjából irreleváns mivel a Traefik közös hálózatra van kötve a helloword service konténereivel, így azon keresztül közvetlen eléri a konténer 80-as portját, nincs szüksége a VM-nek kiajánlott portjára.
- constraints:
    - **node.role == worker**: csak worker node-okra fog telepíteni, manager-ekre nem. Ez mindig követend? példa produkciós környezetben! (ezzel ekvivalens a **node.role != manager**)
    - **node.labels.loadbalancer != true**: Olyan node-ra, aminek van **loadbalancer=true** címkéje nem fog telepíteni. Ilyenb?l ugyebár 1 darab van, a dedikált VM-ünk a load-balancing-ra.

Ahogy azt már említettem, a Traefik service label-ek segítségével azonosítja és konfigurálja azokat a szolgáltatásokat, amikhez load-balancing szolgáltatást kell hogy nyújtson. Ezeket a címkéket a labels szekcióban kell megadni. Alább a minimum címékék:

- **traefik.docker.network**: itt meg kell adjuk azt a load-balancing-re létrehozott overlay hálózatot, amire egyrészr?l a Traefik-et is rákötöttük, másrészr?l az összes load-banace -olandó szolgáltatás konténerei is rá vannak kötve. A fenti példában a **helloword** service konténereit kötjük rá a **balancer-net** overlay hálózatra. Mikor a Traefik a swarm manager-t?l eléri a szolgáltatásokat, akkor csak azokkal foglalkozik, aminek a konténerei rá vannak kötve a 'közös' overlay hálózatra.
- **traefik.port**

- **traefik.frontend.rule=PathPrefixStrip:/hello/**: Na ez itt a legfontosabb. A **traefik.frontend.rule** azt mondja meg, hogy hol legyen elérhet? a load-balancer-t szolgáltatás. A traefik.frontend.rule lehetséges paramétereit itt nézzük meg: https://docs.traefik.io/basics/#frontends. A **PathPrefixStrip**-el megadunk egy path prefix-et. Ha a load-balancer domain neve után olyan URL-t írunk a böngész?be, ami az itt megadott prefixel kezd?dik, akkor a Traefik azt fogja hátraküldeni a konténereknek amit a path-prefix után írtunk, de a path-perfix-et le fogja róla vágni. Pl a http://example.com/hello/index.php URL-b?l a tutum konténerek csak az index.php-t fogják megkapni. Ahogy láttam a válaszban natívan kiegészíti a resource URL-eket az itt megadott path-prefix-el. Tehát ha volt egy **<img src=logo.png/>** akkor abból **<img src=hello/logo.png/>** lesz.
- **traefik.backend.loadbalancer.sticky**: be lehet kapcsolni vele a sticky session-t. Tulajdonképpen ezért csináltuk az egészet, hogy legyen sticky session-ünk. Http session tartás nélkül a natív docker swarm Layer 4 load balancer is kiválóan használható.

Körülbelül 40 féle címkével vezérelhetjük a load balancer m?ködését, itt a teljes lsita: https://docs.traefik.io/configuration/backends/docker/

Hozzuk létre az új szolgáltatást (stack-et) a docker stack deploy paranncssal.

```
# docker stack deploy -c helloworld.yml helloservice
Creating service helloservice_helloworld
```

### Mi jött létre

```
# docker stack ls
NAME                SERVICES
helloservice        1
```

```
# docker stack ps helloservice
ID                  NAME                        IMAGE                        NODE                DESIRED STATE       CURRENT STATE
ygigsppmqp1j        helloservice_helloworld.1   tutum/hello-world:latest     worker0             Running             Running about a minute ago
kv5lvg2g79jn        helloservice_helloworld.2   tutum/hello-world:latest     worker1             Running             Running 59 seconds ago
gc1yqgshnhw3        helloservice_helloworld.3   tutum/hello-world:latest     worker2             Running             Running 59 seconds ago
rgoc8uzt354v        helloservice_helloworld.4   tutum/hello-world:latest     worker1             Running             Running 59 seconds ago
sofw5w45qyco        helloservice_helloworld.5   tutum/hello-world:latest     worker2             Running             Running 59 seconds ago
```

Látható, hogy ahogy kértük, csak worker node-ra telepítette a szolgáltatást, és a loadbalancer nev? node-ra nem rakott egy konténert sem.

```
# docker service ls
ID                  NAME                    MODE                REPLICAS            IMAGE                        PORTS
w31aon1cwm34        helloservice_helloworld   replicated          5/5                 tutum/hello-world:latest     *:30004->80/tcp
04cj4vdsu0qu        loadbalancer              replicated          1/1                 traefik:latest               *:80->80/tcp, *:8080->8080/t
```

Listázzuk ki, hogy milyen hálózatokban van interfésze helloservice-hez tartozó konténereknek. Láthatjuk, hogy a 10.0.2.0/24 és a 10.255.1.0/24-os hálózatokban van tagsága.

```
 # docker service inspect --format='{{.Endpoint.VirtualIPs}}' helloservice_helloworld
 [ {mzwld5ddadk6tcpio8ytkyhgg 10.255.1.10/16}
   {o4rhm35gkh24cd25rdt7hsm62 10.0.2.11/24}]  <<<<<
```

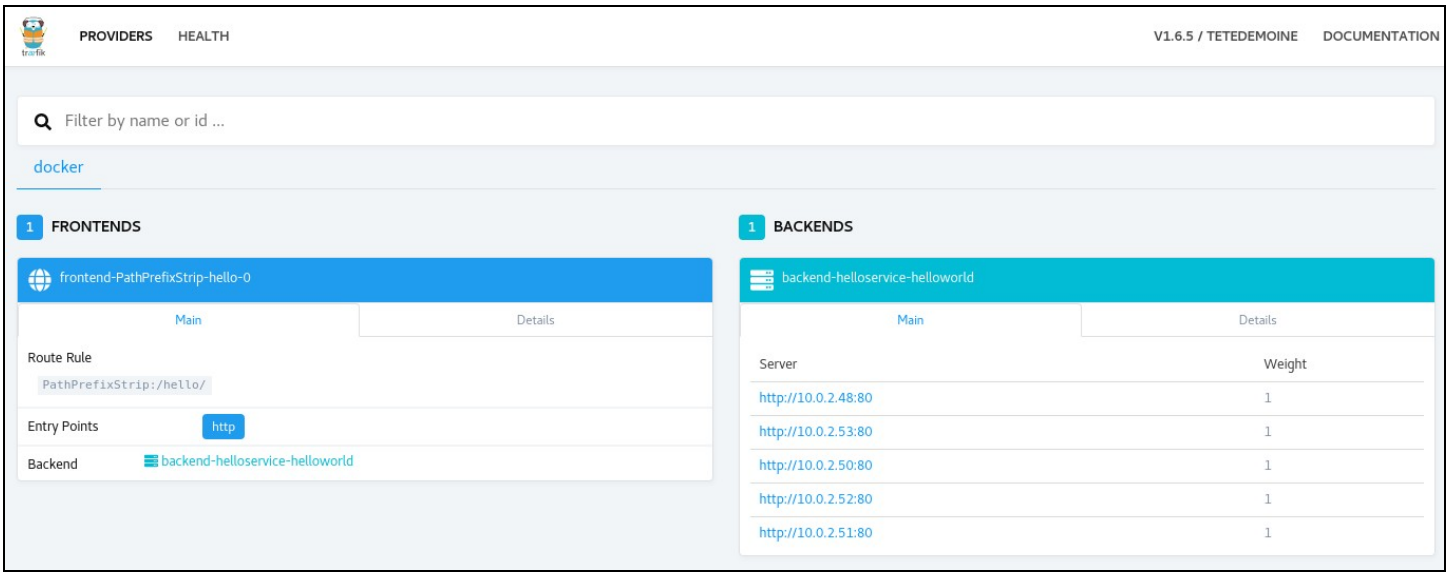
Nézzük meg melyik hálózat micsoda. Ahogy annak lennie kell, az egyik a balancer-net, ami egy közös overlay hálózat a load balancer szolgáltatással. A másik meg a swarm beépített ingress hálózata. Azt most nem fogjuk használni (ezen fut a Layer 4 IPVS).

```
# docker network inspect balancer-net | grep Subnet
                    "Subnet": "10.0.2.0/24",
# docker network inspect 'ingress' | grep Subnet
                    "Subnet": "10.255.0.0/16",
```

## Load balance test

### Traefik konzol a végpontokkal

Els?ként nézzük meg, hogy a Traeifk konzolon megjelent e az új szolgáltatás: http://192.168.42.42:8080/dashboard/




**Frontends**: A frontedns listában megjelent a frontend-PathPrefixStrip-hello-0 nev? szolgáltatás. Ez azt a szabályt tartalmazza, ami megmondja, hogy melyik back-ends-re kell továbbítani a kérést. Három f? komponense van:

- **Route Rule: PathPrefixStrip:/hello/** - ez maga a szabály, amit a compose fájlban megadtunk a label-ek segítségével. Ezt mondja meg, hogy ha olyan URL érkezik a Traefik-hez ami hello/-val kezd?dik, akkor azt irányítsa a megadott backend-hez, úgy hogy a hello/-t levágja róla
- **Entry Points: http** - http-n és nem https-en fogad kéréseket a frontend
- **Backend: backend-helloservice-helloworld** - Ha a szabály teljesül, akkor erre a backend szolgáltatásra kell küldeni a kérést.

**Backends**: A backend listában egy darab szolgáltatás van: backend-helloservice-helloworld. Ebben a main fülön fel vannak sorolva a szolgáltatáshoz tartozó végpontok. Itt láthatjuk mind az 5 konténert, ami a hellword docker stack-hez tartozik. Láthatjuk az IP címüket, amiket a balancer-net overlay hálózaton kaptak. A Details fülön három értéket láthatunk:


- **Method: wrr** - Ez az alapértelmezett load-balancer algoritmus (Weight Round Rubin, részletek itt: https://docs.traefik.io/basics/)
- **Stickiness: true** - Ezt címkékkel mi adtuk meg
- **Cookie Name: _TRAEFIK_BACKEND** - Ezt nem adtuk meg külön label-el, ez az alapértelmezett session süti név, ezzel tartja fent a sticky session-t.

## Böngész? teszt

írjuk be a böngész?be a load-balancer VM 'publikus' IP címét, a /hello path-al: http://192.168.42.42/hello/
Ekkor véletlen szer?en be kell hogy essünk valamelyik tutum konténerre.


Látható, hogy kiírta a konténer ID-ját. Nézzük meg, hogy mi van a háttérben.

Els? lépésként nézzük meg mi van a **helloservice** egyesített log-jában:

```
# docker service logs -f helloservice_helloworld
 helloservice_helloworld.3.w913y7d9pnib@worker2     | 10.0.2.18 - - [30/Jul/2018:07:34:19 +0000] "GET .."-" "Mozilla/5.0 (X11; Fedora; Linux x
```

Láthatjuk, hogy a w913y7d9pnib id-jú szolgáltatás replika kapta meg a kérést.

**Note**
A szolgáltatás ID nem egyenl? a konténer ID-val!. A képerny?n a konténer ID volt kiírva, a log-ban a replika ID-ját látjuk

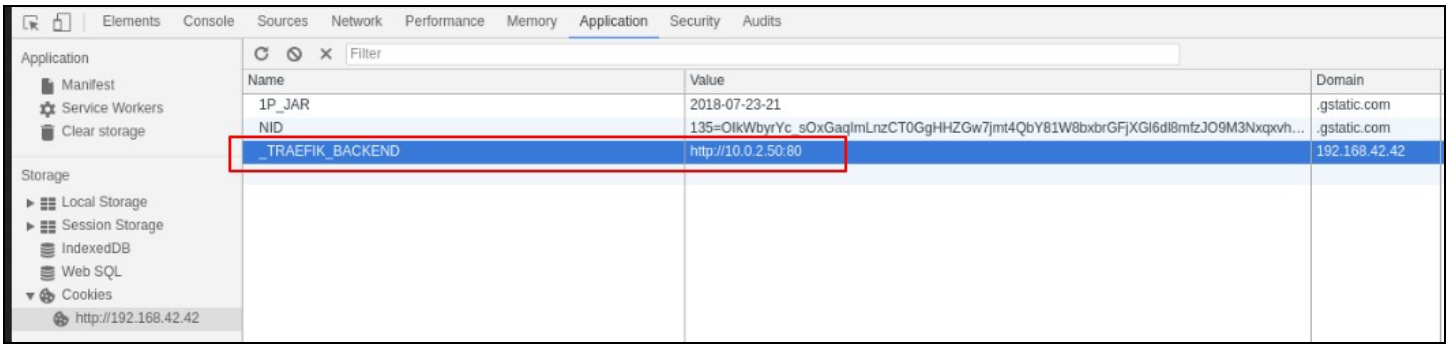
Keressük meg hozzá a konténert. Ehhez a docker insect paranccsal nézzük meg a replika részleteit.

Fontos, hogy itt az inspect szintén nem a konténerre vonatkozik. Swarm mode-ban nem lehet konténer specifikus utasításokat kiadni (max azon a VM-en ahol a konténer fizikailag van). Itt az Insect a szolgáltatás példányra vonatkozik!

```
# docker inspect w913y7d9pnib
     ....
       "ID": "w913y7d9pnibnsp6dso3bpqbl",
         ...
       "ContainerStatus": {
              "ContainerID": "2aadf0b32b329a62d1bc916ce3dd67264f8a47d2042f6f4070ee979cfb271bcd",
        ....
        "Addresses": [
                  "10.0.2.50/24"
            ...
```

Látható, hogy a **ContainerID** szekcióban kiírta a konténer valódi ID-ját, ami megegyezik azzal ami a képerny?n megjelent. Ha most ssh-val belépnénk a worker2 VM-re ahol ez a konténer fizikailag telepítve van, ott már megtalálnánk a **docker ps** paranccsal a konténert.

Ha frissítjük a böngész?t, akkor is ezen a lábon maradunk, tehát a sticky session m?ködik!
Nézzük meg a site-hoz tartozó sütit a developer-tools-ban:




Látható, hogy a **_TRAEFIK_BACKEND** nev? session süti lerakásra került és az értéke a b32 **konténer IP címe (10.0.2.50) a balancer-net overlay hálózaton, amit a fenti** docker inspect **eredményében is láthatunk. Ez az IP cím szerepel a Traefik konzolon is a backend server listában.**

# Statefull load-balancing with NGINGX

https://www.nginx.com/blog/docker-swarm-load-balancing-nginx-plus/#nginx-demo

# Docker Flow Proxy

https://proxy.dockerflow.com/