

Szybki Kurs: HAProxy LB w Dockerze — Zadania, Ćwiczenia i Analiza Część 3

Cel kursu

Celem kursu jest nauczenie Sebastiana pracy z HAProxy jako load balancerem HTTP oraz konfiguracji środowiska w Docker Compose z wieloma backendami.

Wstęp

Load balancer to mechanizm, który równomiernie rozkłada ruch sieciowy pomiędzy wiele serwerów, aby zwiększyć wydajność i dostępność aplikacji. Dzięki niemu system może obsługiwać więcej użytkowników, a awaria jednego serwera nie powoduje przerwy w działaniu usługi.

Najpopularniejszym mechanizmem jest **roundrobin (WAŻNE, zapamiętaj)** czyli kierowanie kolejne zapytania po kolei na każdy serwer w pętli. Nie uwzględnia on obciążenia serwerów, więc każdy dostaje taką samą liczbę żądań, niezależnie od swojej aktualnej wydolności.

Alternatywy:

- **Weighted Round-Robin** – przydziela ruch proporcjonalnie do „mocy” serwerów.
- **Least Connections** – kieruje ruch do serwera z najmniejszą liczbą aktywnych połączeń.
- **Least Response Time** – wybiera serwer z najniższym czasem odpowiedzi.
- **IP Hash / Consistent Hashing** – ten sam klient trafia zawsze na ten sam serwer (przydatne przy sesjach).
- **Random** – wybór losowego serwera, czasem z wagami.

Przykład – wycinek haproxy.cfg:

```
frontend http_in                # frontedn nazwa
    bind *:80                   # nasługuje na wszystkich ip na porcie 80
    default_backend hostgui_nodes # przekazuje na backend o nazwie

backend hostgui_nodes           # backend nazwa
    balance roundrobin          # mechanizm balancingu
    option httpchk GET /
    server hostgui1 192.168.1.20:80 check # server1
    server hostgui2 192.168.1.21:80 check # server2 ...
    server hostgui3 192.168.1.23:80 check # server3 ...
    server hostgui4 192.168.1.24:80 check # server4 ...

# Nowe serwery dodajesz
# server twoja-nazwa nazwa-dns/ip:PORT check
```

Haproxy jak większość balancerów swoją konfigurację opiera na podziale na:

Frontend – gdzie ustawiamy np. na jakim ip i porcie działa, jaki ma certyfikat ssl (jeśli chcesz usytawić httpd), przekierowania, itp.

Backend – gdzie ustalasz na jakie i ile serwerów ma kierować ruch

Zadanie 3.1

W nowym katalogu stwórz plik docker-compose.yaml

Uruchom 1 serwis z obrazem `spidero/hostgui`, jest w nim aplikacja webowa uruchomiona na serwerze nginx działająca wewnątrz kontenera na porcie 80

Wystaw ten port tak żebyś mógł połączyć się przeglądarką do tego obrazu

WAŻNE!!! w dockerze nazwy serwisów są również jego nazwa dnsowa

Czyli np. jeśli nazwiesz serwis xyz, to z innego kontenera możesz używać tej nazwy zamiast ip (które się zmienia) do pinga, dostępu itp.

Potrzebne info do konfiguracji haproxy

uruchom serwis komenda

docker compose up

lub

docker compose up -d

Powinieneś zobaczyć coś takiego:

Kurs Dockera: Sebastian, zadanie z nginx/LB

Informacje o kontenerze

Parametr	Wartość
Hostname	a2ef66b9d2ea
Adres IP	172.27.0.2
Data uruchomienia	Sun Dec 7 19:34:04 UTC 2025
Wersja Nginx	nginx version: nginx/1.29.2
System	Alpine Linux v3.22

Zapisz wynik polecenia **docker ps**

Zapisz plik docker-compose.yaml w gitcie w katalogu zadanie-3.1

Zatrzymaj serwis komendą **docker compose down** (musisz być w katalogu z plikiem docker-compose.yaml)

Zadanie 3.2

Na podstawie zadanie 3.1 rozszerz swój plik docker-compose.yaml tak aby usyskać 4 serwisy oparte o ten obraz `spidero/hostgui`

uruchom serwisy komenda

docker compose up

lub

docker compose up -d

Zapisz wynik polecenia **docker ps**

Zapisz wynik polecenia **docker compose stats --no-stream**

Zapisz plik docker-compose.yaml w gitcie w katalogu zadanie-3.2

Spróbuj połączyć się przeglądarka każdy serwisów, zwróć uwagę co się zmienia w aplikacji. Zapisz, opowiedz

Zatrzymaj serwis komendą **docker compose down** (musisz być w katalogu z plikiem docker-compose.yaml)

Zadanie 3.3

Na podstawie zadanie 3.2 rozszerz swój plik docker-compose.yaml o kolejny serwis, tym razem będzie to load balancer haproxy

Dla ułatwienia wklejam serwis

```
services:
# TODO: dopisz serwisy backendowe hostgui1-hostgui4

haproxy:
  image: haproxy
  container_name: haproxy
  ports:
    - "8085:80"          #port na którym jest frontend
    - "9001:9001"        #port panelu admina haproxy
  volumes:
    - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg:rw
  depends_on:
    - nazwa_serwisu_z_aplikacja_hostgui
```

W katalogu dodaj plik **haproxy.cfg** o zawartości:

```
global
    log stdout format raw local0
    maxconn 2000

    # ♦ Runtime socket dla Data Plane API (współdzielony przez volume)

    stats socket /tmp/haproxy.sock mode 660 level admin
    stats timeout 30s

    user root
    group root

defaults
    log        global
    mode       http
    option      httplog
    option      dontlognull
    timeout     connect 5s
    timeout     client  30s
    timeout     server  30s
    retries     3

frontend http_in
    bind *:80
    default_backend hostgui_nodes

backend hostgui_nodes
    balance roundrobin
    option httpchk GET /
    server hostgui1 hostgui1:80 check
    server hostgui2 hostgui2:80 check
    server hostgui3 hostgui3:80 check
    server hostgui4 hostgui4:80 check

# ♦ GUI statystyk
listen stats
    bind *:9001
    mode http
    stats enable
    stats uri /
    stats refresh 5s
    stats show-legends
    stats show-node
    stats auth admin:admin123    # login: admin / hasło: admin123
    stats admin if TRUE
```

To musisz dopasować

Login, hasło do gui
admina

Zapisz wynik polecenia `docker ps`

Zapisz wynik polecenia `docker compose stats --no-stream`

Zapisz plik `docker-compose.yaml` w gitcie w katalogu `zadanie-3.2`

Spróbuj się połączyć przeglądarka na porty 8085 i 9001, zwróć uwagę co się zmienia, opisz, omów

Wklej screany na git

Powinienes zobaczyć cos takiego:

na porcie 9001

HAProxy version 3.3.0-7832fb2, released 2025/11/26

Statistics Report for pid 100 on 50fad14fcabb

> General process information

pid = 100 (process v1, rproc = 1, rthread = 0)
uptime = 0.00 (0.00/0.00), warnings = 0
system memsize (memsize) = 22.500 MB, ulimit = 4096
maxsock = 4096, maxconn = 2000, reached = 0, maxpipes = 0
current curbs = 0, current objcs = 0, curbs rate = 1/sec, bk rate = 0.815 Mbps
Running tasks: 0/0 (0 ready), sbr = 100%

active UP
active UP going down
active DOWN going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SCOT STOPPED for maintenance
Note: "NO LB T'CRARY" = UP with load-balancing disabled

[X] Action processed successfully.

Info on		Queue						Sessions						Bytes				Denied				Errors				Warnings		Status
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LibTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis						
	Frontend				0	0	-	0	0	0	2 000	0				0	0	0	0	0			OPEN					

hostipg nodes		Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status				
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LibTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis		
<input type="checkbox"/>	hostipg1	0	0	-	0	0	-	0	0	0	-	0	0	7	0	0	0	0	0	0	0	0	23m26s UP	LTC
<input type="checkbox"/>	hostipg2	0	0	-	0	0	-	0	0	0	-	0	0	7	0	0	0	0	0	0	0	0	23m26s UP	LTC
<input type="checkbox"/>	hostipg3	0	0	-	0	0	-	0	0	0	-	0	0	7	0	0	0	0	0	0	0	0	23m26s UP	LTC
<input type="checkbox"/>	hostipg4	0	0	-	0	0	-	0	0	0	-	0	0	7	0	0	0	0	0	0	0	0	23m26s UP	LTC
	Backend	0	0	-	0	0	-	0	0	0	200	0	0	7	0	0	0	0	0	0	0	0	23m26s UP	

Choose the action to perform on the checked servers:

Stats		Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status
		Cur	Max	Limit	Cur	Max	Limit	Total	LibTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr		
	Frontend				1	1			1	2	2 000	1		0	0	0	0	0		OPEN
	Backend	0	0	-	0	0	-	0	0	0	200	0	0	0	0	0	0	0	0	23m26s UP

na porcie 8085 (polecam użyć tryb incognito żeby ominąć problem z cachem)

Kurs Dockera: Sebastian, zadanie z nginx/LB

Informacje o kontenerze

Parametr	Wartość
Hostname	a2ef66b9d2ea
Adres IP	172.27.0.2
Data uruchomienia	Sun Dec 7 19:34:04 UTC 2025
Wersja Nginx	nginx version: nginx/1.29.2
System	Alpine Linux v3.22

odśwież parę razy strone, co się zmienia i dlaczego?

Ćwiczenia praktyczne

- Odśwież stronę 10 razy i sprawdź round robin.
- Obserwuj backendy w panelu statystyk.
- Zatrzymaj hostgui3 i sprawdź status DOWN.
- Wprowadz któryś host w tryb maintainane