# 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 wszystych 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 swoja konfiguracje opiera na podziale na:
**Frontend** – gdzie ustawiamy np. na jakim ip i porcie działa, jaki ma certyfikat ssl (jeśli chcesz usytawić httpd), przekirowania, itp.

**Backend** – gdzie ustalasz na jakie i ile serwerów ma kierowac 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 wew konenera na porcie 80

Wystaw ten port tak żebys mogł połączyć się przeglądarka do tego obrazu

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

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

**Pottzebne info do konfiguracji haproxy**

uruchom serwis komenda

**docker compose up**
lub
**docker compose up -d**

Powinieneś zobaczyć coś takiego:

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 swoj 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

Spróbój połączyć się przegladarka każdy serwisów, zwróć uwage 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

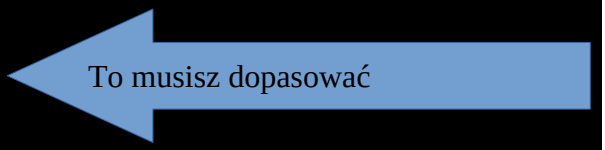
Na podstawie zadanie 3.2 rozszerz swoj plik docker-compose.yaml o kolejny serwis, tym razem będzie to load balancer haproxy

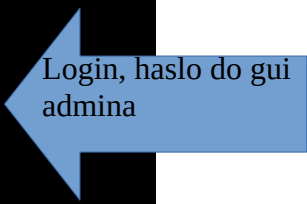
Dla ułatwienia wklejam serwis

```
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
```



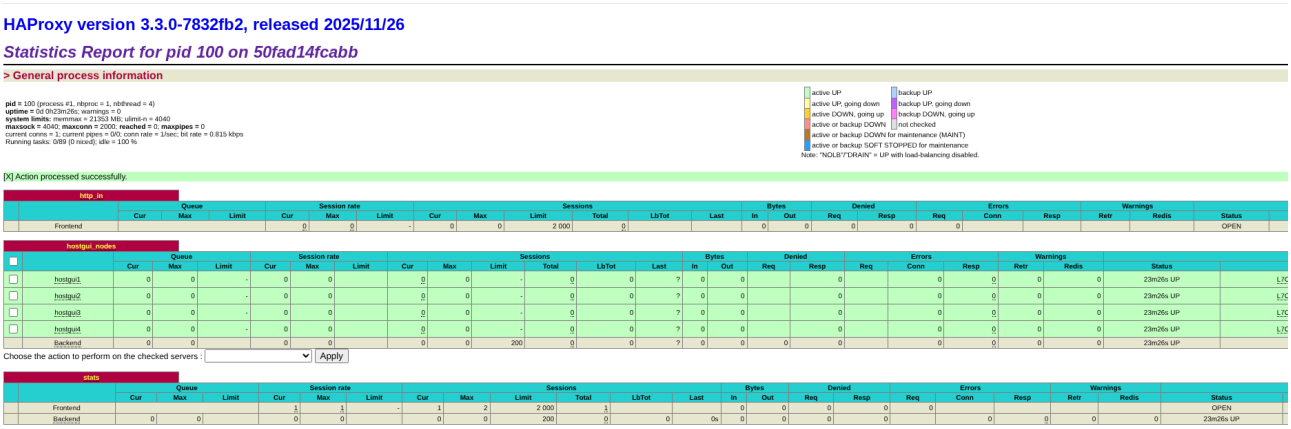

Zapisz wynik polecenia docker ps
Zapisz wynik polecenia `docker compose stats --no-stream`
Zapisz plik docker-compose.yaml w gitcie w katalogu zadanie-3.2

Sprobój się połączyć przegladarka na porty 8085 i 9001, zwróć uwage co się zmienia, opisz, omów

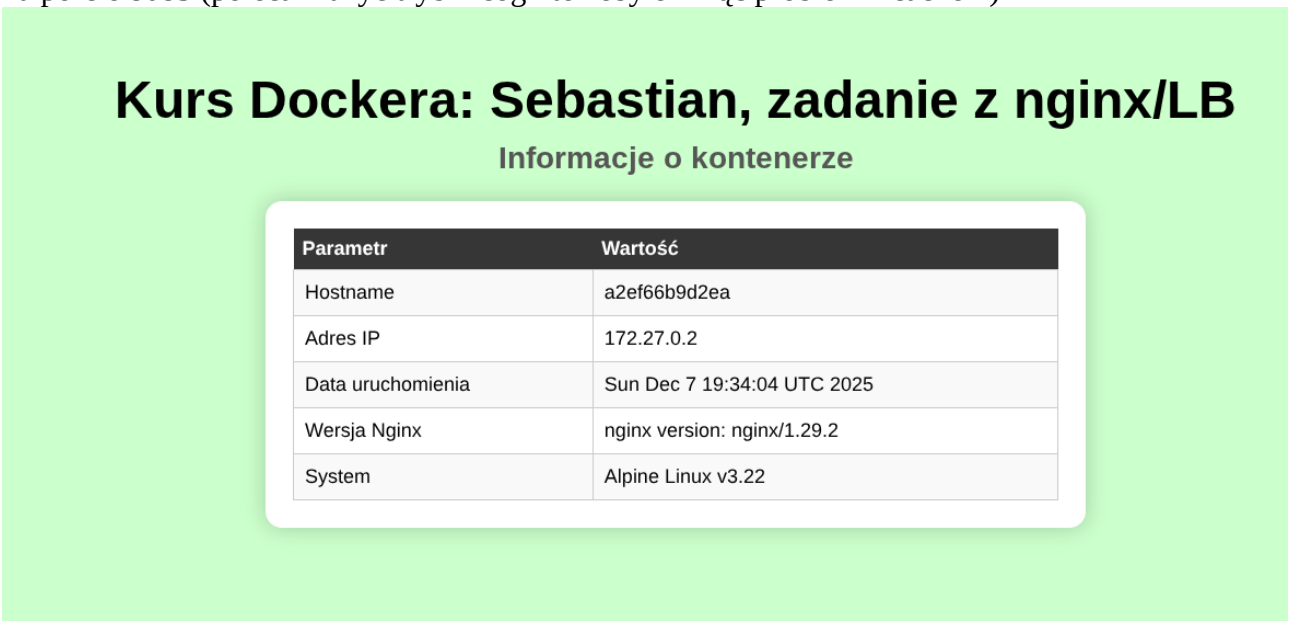
Wklej screany na git

Powinienes zobaczyć cos takiego:

na porcie 9001




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




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 kótrys host w tryb maintanane