SCD
Laboratorul #6 Cache și consistență distribuită (Redis)

Versiunea 1

Responsabili: **Dorinel FILIP**

Obiectivele laboratorului

În urma parcurgerii laboratorului, studenții:

- Vor identifica nevoi de păstrare a consistenței în servicii web multi-server (cu mai mulți workeri de tip backend);
- Se vor familiariza cu Docker Swarm și cu posibilitatea de a porni mai multe instanțe ale unui serviciu folosind Docker;
- Se vor familiariza cu tehnologia Redis și vor realiza primitivele necesare unui sistem de limitare a ratei cererilor (rate-limiting).

Noțiuni teoretice / Pași pentru pregătirea laboratorului

Folosind acest ghid sau documentația oficială, familiarizați-vă cu Docker Swarm și inițiați (folosind **docker swarm init**) un cluster de Docker Swarm format doar din computerul vostru, fără alți workeri.

Redis este o tehnologie de tipul in-memory data store folosită, datorită timpului foarte mic de răspuns, în special pentru sincronizarea multiplelor replici ale aplicațiilor distribuite.

În cadrul acestui laborator, vom rula (folosind Docker Swarm) mai multe replici ale acestei aplicații web și vom implementa un mecanism distribuit de numărare a request-urilor făcute în ultimele 15 secunde, folosind Redis.

Exerciții

Exercițiul 1

Plecând de la exemplul de documentația Docker Compose pe care-l puteți descărca și de aici, modificați aplicația astfel încât:

- Să afișeze și hostname-ul containerului în care oferă acel răspuns (hint: `https://docs.python.org/3/library/socket.html#socket.gethostname`);
- Să afișeze accesările din ultimele 10 secunde, **sliding_window** (hint: `https://www.peakscale.com/redis-rate-limiting/`).

Pentru simplitate, vom ignora cazul în care 2 sau mai multe request-uri ajung la același marcaj de timp și eventualele clock-skew.

Exercițiul 2 - Docker Swarm și mai multe replici

Rulați aplicația de la Exercițiul 1 folosind **Docker Swarm** și modificați fișierul YML astfel încât să ruleze 3 replici ale serviciului web (hint).

Observație: Pentru a putea rula cu Docker Swarm, va trebui să eliminați opțiunile imcompatibile din YML și să dați un nume imaginii.

Accesați de mai multe ori pagina. Ce observați cu privire la hostname-ul containerului care servește request-ul?

Exercițiul 3 - Docker Compose/Stack elegant

Rulați aplicația de la Exercițiul 1 folosind **Docker Swarm** și duceți la îndeplinire următoarele cerințe:

- Adăugați o parolă pentru serverul de Redis;
- Adăugați variabile de mediu pentru:
  - Datele necesare conectării de la serviciul web la serverul de Redis (HOST, PORT, PAROLA);
  - Configurarea perioadei de timp în care se va face numărarea request-urilor.
- Demonstrați că variabilele de mediu sunt folosite schimbând numele celor 2 servicii din YML și actualizând corespunzător variabilele.

Exercițiul 4 - Publicarea imaginilor într-un depozit de imagini

Creați-vă un cont pe Docker Hub și publicați imaginea cu numele **username/scd6** și tag-ul **latest**.

Hint: **docker login** + **docker push**.

Comenzi utile pentru rezolvarea laboratorului

```
# Cu Docker Compose
docker compose up --build             # Porneste stack-ul de servicii si face build imaginilor
docker compose up -d                  # Porneste stack-ul in background
docker compose down                   # Opreste stack-ul

# Cu Docker Swarm
docker swarm init                     # Initializeaza un nou cluster de Docker Swarm
docker build . -t scd_web             # Face build imaginii definite de Dockerfile-ul din folderul curent cu numele scd_web
docker stack ls                       # Listeaza stack-urile care ruleaza in Docker Swarm
docker stack deploy -c stack.yml scd  # Porneste un nou stack cu numele scd
docker service update scd_web --force # Forteaza actualizarea serviciului web din stack-ul scd
docker stack rm scd                   # Sterge stack-ul scd din Docker Swarm
docker service ps scd_web --no-truc   # Arata starea task-urilor serviciului web din stack-ul scd
```