Fase 4 - Activitat 7.7: Distribució de carrega i escalabilitat.

0- Identificació del grup i activitat:

Curs: ASIX2
Projecte: PJ9 DevOps i Cloud Computing
Fase: 4
Activitat: 7.7
Grup/Individual: Individual
Membres/Alumne:

1- Introducció i objectius de l'activitat 7.7

El **primer objectiu** es treballar la **distribució de carrega** entre múltiples contenidors que treballen en xarxa. Quan una apliació s'ha desplegat per mitjà de contenidors Docker:

- Per podar donar el servei de manera adequada a tots els usuaris, normalment no es desplega un únic contenidor, sino que es depleguen multiples contenidors.
- Si hi ha molta demanda d'accés a l'aplicació, s'hauran de posar en marxa molt contenidors que treballin en xarxa i coordinats per poder repartir-se la carrega de treball.
- La distribució o balanceig de carrega és una tècnica que permet que diversos contenidors es distribueixin la carrega de treball entre ells si estan en xarxa i tenen algun sistema que permeti coordinar-los.
- Per poder distribuir la carrega entre diversos contenidors, en caldrà un contenidor especial que faci de Proxy HTTP. Un proxy HTTP fa la feina de distribuir les peticions d'accés a l'aplicació que van arribant entre els diversos contenidors de l'aplicació en funció de la seva carrega.

El **segon objectiu** es treballar el concepte d'**escalabilitat**:

- L'escalabilitat és la capacitat de posar en marxa o aturar multiples contenidors que executin una mateixa aplicació en funció de la demanda d'accés a l'aplicació.
- Si la demanda augmenta, s'han de poder posar en marxa més contenidors que executin una mateixa aplicació i, si disminueix la demandam se n'hauria de reduir la quantitat de contenidors en marxa.
- Tot això s'ha de fer de manera ràpida, fàcil, eficient i transparent als usuaris.

Utilitzant l'eina **docker compose** és fàcil:

- Afegir escalabilitat a una aplicació desplegada per mitjà de contenidors dockers.
- Afegir distribució de carrega entre els contenidors de l'aplicació.

Així doncs, dins d'aquesta activitat:

- Desplegarem multiples contenidors que executin una mateixa aplicació web d'una manera fàcilment escable.
- Es distribuirà la carrega entre els contenidors de manera coordinada i sincronitzada afegint al nostre projecte un tipus especial de contenidor que actuarà com a Proxy HTTP.
- Utilitzarem docker compose poder fàcilment desplegar un sistema que treballi amb distribució de carrega i que sigui escalable.

2- Distribució de carrega a una aplicació utilitzant un proxy HTTP amb nginx

2.1- Creant els fitxers Dockerfile, docker-compose.yml i nginx.conf del projecte

a) Accedeix a la màquina virtual que vas crear dins de l'activitat **pj9f4a7.6** i accedeix al directori **codis**.

b) Ara, modifica **Dockerfile** fent que tingui el següent contingut:

```
# Imatge base a partir de la qual crearem la nostra
FROM php:8.4-apache
# Directori per defecte de treball
WORKDIR /var/www/html
# Copia la carpeta app dins "." de contenidor, a on "." és el WORKDIR
COPY app .
# Exposa el port 80 del contenidor
EXPOSE 80
```

c) A continuació, modifica **docker-compose.yml** fent que tingui ara el següent contingut:

```
services:
  sLlaunes: # Servei donat per l'aplicació
    image: illauna:1.1 # Imatge a construir per crear els contenidors
    build: . # Lloc a on es troba el Dockerfile i app
    environment:
      - HOST_NAME=pj9f4a76-dacomo.fjeclot.net
    expose:
      - 80 # Exposa el port 80 dels contenidors però no els publica a la màquina virtual
    networks:
      - xPj9f4a7.7 # Nom de la xarxa dins de docker-compose.yml

  sDistrubeixCarrega: # Servei Proxy HTTP per fer distribució de carrega
    image: nginx:latest # Imatge de contenidor amb el proxy HTTP que es baixarà
    container_name: cProxyHTTP # Nom del contenidor amb el proxy HTTP
    volumes: # Compartint nginx.conf de la màquina virtual amb el contenidor
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on: # Si SLlaunes no funciona, sDistribueixCarrega tampoc.
      - sLlaunes
    ports: # Exposa i pública el port 80 del proxy HTTP al port 8000 de la màquina virtual
      - 8000:80
    networks:
      - xPj9f4a7.7

networks:
  xPj9f4a7.7:
    name: pj9f4a7.7 # Nom de la màquina quan fem docker networks ls
    driver: bridge
```

d) Dins de la carpeta **codis** de la màquina virtual crea el següent arxiu de nom **nginx.conf**:

```
user  nginx;

events {
    worker_connections   1000;
}
http {
        server {
              listen 80;
              location / {
                proxy_pass http://sLlaunes:80;
              }
        }
}
```

NOTA: Si voleu entendre com funciona aquesta configuració, a l'**Annex-1** que hi ha al final de la pràctica està tot explicat.

2.2- Instanciant una xarxa de 5 contenidors amb l'aplicació i un servidor proxy

a) Executa dins del directori a on es troba **docker-compose.yml** l'ordre:

```
docker compose up -d --scale sLlaunes=5
```

b) Comprova que:
- S'han creat **5** contenidors de l'aplicació d'operacions matemàtiques de nom **codis-sLlaunes-1** a **codi-sLlaunes-5**.
- S'ha creat un contenidor amb el distribuidor de carrega **nginx** de nom **cProxyHTTP**
- S'ha creat una xarxa de dockers de nom **pj9f4a7.7**
- Que els 5 contenidors de l'aplicació i el distribuidor de carrega estan a la mateixa xarxa. Executa: **docker network inspect pj9f4a7.7 | grep Name**

2.3- Comprovació de funcionament de l'aplicació

a) Comprova l'adreça IP de la màquina virtual.

b) Accedeix al port **8080/tcp** de la màquina virtual amb el navegador des de la màquina física utilitzant l'adreça IP trobada a l'apartat anterior i comprova que l'aplicació funciona correctament.

c) Comprova que l'aplicació funciona completament.

d) Comprova que el balanceig de carrega funciona mirant l'identificador del contenidor al qual accedeixes cada cop que refresques la pàgina inicial.

2.4- Aturant i esborrant la xarxa i contenidors creats amb docker-compose

a) Executa dins de la carpeta **codis** de la màquina virtula l'ordre: **docker compose stop** i comprova que:
- S'aturen els contenidors
- Els contenidors i la xarxa no s'esborren
- L'aplicació deixa de funcionar

b) Executa dins de la carpeta **codis** de la màquina virtual l'ordre: **docker compose start** i comprova que:
- S'inicien els contenidors
- L'aplicació torna a funcionar

c) Executa dins de la carpeta **codis** de la màquina virtual les ordre:

```
docker compose stop
docker compose down
```

i comprova que els contenidors, la xarxa i el servei han desaparegut.

2.5- Comprova el balanceig de carrega des del terminal

a) Executa dins de a carpeta **codis** de la màquina virtual l'ordre: **docker compose up --scale sLlaunes=5** i comprova que per pantalla surten els missatges de creació i arrancada dels contenidors de l'apliacació i del proxy. És **important** que **sigui sense -d** per veure els contenidors treballant a la pantalla, **en primer terme**.

b) Accedeix amb el navegador de la màquina física a l'aplicació utilitzant l'adreça IP de la màquina virtual i comprova que l'aplicació funciona. Comprova a quin contenidor s'ha accedit per mostrar l'aplicació.

c) Comprova que si recarregues la pàgina inicial, va canviant el contenidor al qual ens connectem per accedir a l'aplicació passant sempre pel proxy.
d) Atura els contenidors de l'aplicació amb **Ctrl+c**. Deprés, esborra'ls.

3- Escalant i desescalant l'aplicació

a) Inicia l'aplicació amb **3** contenidors**.** Executa:

docker compose up --scale sLlauna=3 -d

b) Comprova que s'executen **3** dockers de l'aplicació amb l'ordre: **docker compose ps -a**

c) Incrementa la quantitat de contenidors que executen l'aplicació sense aturar-la de **3** a **12**. Executa:

docker compose up --scale sLlaunes=12 -d

d) Comprova que s'executen **12** dockers de l'aplicació amb l'ordre: **docker ps -a**

e) Redueix la quantitat de contenidors que executen l'aplicació sense aturar-la de **12** a **6**. Executa:

docker compose up --scale sLlaunes=6 -d

f) Comprova que s'executen **6** dockers de l'aplicació amb l'ordre: **docker compose ps -a**

4 – Pujant els codis al dipòsit local

a) Si tot funciona correctament, dins de la màquina virtual accedeix directori **codis** i executa:

```
git add .
git commit -m "Estat del projecte com van quedar després de fer l'activitat pj9f4a7.7"
```

b) Comprova amb **git log** que s'ha afegit al dipòsit la versió actual del projecte.

Lliurament de l'activitat

Part 1 – Balanceig de carrega i accés a l'aplicació
a) Posa en marxa **5** contenidors amb l'aplicació del càlcul del cost de llaunes de begudes i comprova que s'han creat.
b) Des de la màquina física accedeix a l'aplicació i comprova que funciona.
c) Des de la màquina física, refresca diverses vegades la pàgina inicial per comprovar que el balanceig de carrega funciona dins del navegador. Com saps que canvia de contenidor?.
d) Atura l'aplicació (sense esborrar-la) i comprova que l'aplicació deixa de funcionar

Part 2 – Balanceig de carrega i accés a l'aplicació
a) Posa en marxa **5** contenidors amb l'aplicació del càlcul del cost de llaunes de begudes treballant en primer terme.
b) Des de la màquina física accedeix a l'aplicació. Refresca diverses vegades la pàgina inicial per comprovar que el balanceig de carrega funciona dins del navegador. Com saps que canvia de contenidor des del terminal?.
c) Atura l'aplicació (sense esborrar-la) i comprova que ja no està funcionant.

Part 3 – Escalabilitat
a) Posa en marxa l'aplicació amb **12** contenidors. Mostra que s'han creat els **12** contenidors.
b) Passa a treballar amb **3** contenidors. Mostra que ara només hi ha **3** contenidors.
c) Passa a treballar amb **6** contenidors. Mostra que ara només hi ha **6** contenidors.
d) Atura i esborra l'aplicació i comprova que el servei, la xarxa i els contenidors han desaparegut.

Part 4- Pujant els codis al dipòsit local
a) Mostra que tens **2** commits. Els de les activitats **pj9f4a7.6** i **pj9f4a7.7**.

Data límit per obtenir el 100% de la nota: **dimarts 1-12-24** a les **19.10**.

ANNEX 1 – EXPLICACIÓ DE LA CONFIGURACIÓ

a) Es crearà i iniciarà un servei **sDistribueixCarrega** que:

- Posa en marxa un contenidor de nom **cProxyHTTP** amb un servidor **Proxy HTTP** per poder donar servei de distribució de carrega.
- El contenidor es crea a partir de la imatge descarregada **nginx:latest**. que té a dins un servidor **Proxy HTTP** de nom **Nginx**.
- Crea un **volum compartit** per poder modificar **nginx.conf** des de la màquina host i que la modificació quedi en l'arxiu de configuració **nginx.conf** del contenidor **cProxyHTTP**.
- El contenidor exporta el port **80/tcp intern** al port **8000/tcp** de la màquina host. Així dóncs:
  - una connexió a la màquina virtual pel port **8000/tcp** és una connexió al port **80/tcp** del **Proxy HTTP**.
  - Per accedir a l'aplicació hem de connectar-nos a l'adreça IP de la màquina virtual i al port **8000/tcp**.
- El servei no pot iniciar-se si no s'inicia el servei **sLlaunes**.

b) Es crearà i iniciarà el servei **sLlaunes** que:

- Pot posar en marxa multiples contenidors amb l'aplicació. Cada contenidor s'anomenarà **codis-sLlaunes-1**, **codis-sLlaunes-2**, **codis-sLlaunes-3**, etc...
- El contenidor es crearan a partir de la imatge **illauna:1.1** que utilitza **Dockerfile** i **app**.
- Cada contenidor exposarà el seu port **80/tcp** dels contenidors a un port efímer , o sigui, temporal (normalment a partir de **32678/tcp**) de la màquina host (en el nostre cas, la màquina virtual) gràcies al paràmetre **expose**.
- Els contenidors no s'interfereixen entre ells
- En el moment d'establir una connexió:
  - Connexió des de la màquina física al port **8000/tcp** de la màquina virtual.
  - Redirecció del **8000/tcp** de la màquina virtual al **80/tcp** pel qual escolta el proxy **nginx**
  - **nginx** recull la connexió i escull un contenidor (en aquest cas per round robin)
  - **nginx** redirecciona la connexió al seu port **80/tcp** cap el port efímer de la màquina host associat al contenidor escollit.
  - El programari **docker** redirecciona el port efimer cap el port **80/tcp** intern del contenidor.

c) Es crearà una xarxa de contenidors identificada com a **xPj9f4a7.7** dins del fitxer **.yml** i identificada com **pj9f4a7.7** dins de la màquina virtual.

d) Es configurarà el servei Proxy HTTP de manera que**:**

- Faci de distribuidor de carrega del multiples contenidors dins del servei **sLlaunes**.
- El distribuidor de carrega s'encarrega a cada moment de seleccionar el contenidor que ha de respondre a una petició d'accés l'aplicació.
- El distribuidor de carrega i el programari de docker treballaran conjunatament per redirigir les peticions cap el contenidor del servei escollit pel Proxy HTTP.