Fase 4 - Activitat 8.1: Docker en xarxa. Distribució de carrega entre dockers. Docker Compose. Escalabilitat.

0- Identificació del grup i activitat:

Curs: ASIX2
Projecte: GP2 DevOps i Cloud Computing
Fase: 4
Activitat: 8.1
Grup:
Membres:

1- Introducció i objectius de l'activitat 8.1

a) Lectura de les especificacions de l'activitat i creació d'una planficació amb Trello o Gantt
b) Xarxa de contenidors
c) Distribució de carrega entre contenidors
d) Utilització de l'eina docker-compose
e) Escalabilitat

2- Abans de començar

a) Dins del directori **projectes** de la teva màquina físic clona el dipòsit de **Github** que té la següent adreça URL: **https://github.com/globproj2/gp1f4a8.git**. Comprova que s'ha creat una carpeta gp1f4a8 amb la següent estructura:

- Un directori **.git**
- Un directori **codi** amb una aplicació PHP.
- Un directori **vm** amb un fitxer **Vagrantfile**.

b) Crea i posa en marxa una màquina virtual a partir del fitxer **Vagrantfile**. Accedeix a la màquina virtual i comprova que té una adreça IP de la xarxa a la xarxa **eth1** i s'ha creat una carpeta de nom **gp1f4a8/codi** amb dos fitxer **.php**.

c) Dins de la màquina virtual, canvia al directori **/home/vagrant/gp1f4a8**. Crea un fitxer **Dockerfile** amb el següent contingut:

```
FROM php:7.4-apache
WORKDIR /var/www/html
COPY codi .
EXPOSE 80
```

d) Crea una imatge de nom **llaunes** i versió **5.0**.

e) Crea un contenidor:

- Nom: **test**
- Variable d'entorn: **HOST_NAME=$(hostname --fqdn)**
- Treballa en 2n terme i permet accedir a l'interior del contenidor via terminal
- Exportació de ports: **80/tcp** del contenidor a **80/tcp** de la màquina virtual
- Imatge: **llaunes:5.0**

f) Comprova que des de la màquina física pots accedir a l'aplicació i funciona correctament. Comprova que l'aplicació sempre mostra el nom del servidor i que els càlculs també funcionen.

g) Si l'aplicació funciona, puja la nova versió **llaunes:5.0** al teu dipòsit de **DockerHub**.

h) Finalment, atura i esborra el contenidor **test**.

3- Distribució de carrega entre múltiples contenidors executant la mateixa aplicació sobre un únic servidor. Eina docker-compose. Servidor proxy nginx

3.1- Creant el fitxer docker-compose.yml

a) Dins de la carpeta **gp1f4a8** crea el següent arxiu de nom **docker-compose.yml**:

```
version: '3'

services:
  tonica:
    image: llaunes:5.0
    expose:
      - "80"
    environment:
      - HOST_NAME=gp1f4a8.fjeclot.net

  lb:
    image: nginx:latest
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - tonica
    ports:
      - "80:8000"
```

NOTA: Les separacions i espais són molt importants quan es treballa amb fitxer de configuració forrmat **.yml** o **.yaml**.

b) Què fa aquesta configuració?:

- Crearà un contenidor de nom **tonica**:
  - A partir de la imatge **app_llaunes:5.0**.
  - Exposarà el port **80/tcp** del contenidor **tonica** a un port efímer (normalment a partir de **32678/tcp**) de la màquina host (en el nostre cas, la màquina virtual).
- Crearà un contenidor de nom **nginx**:
  - Nom: **lb**
  - A partir de la imatge **nginx:latest**.
  - 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.
  - El contenidor exporta el port **8000/tcp intern** al port **80/tcp** de la màquina host.
  - El contenidor **nginx** no s'inicia si el contenidor **tonica** no està en marxa.
- Crearà una xarxa de contenidors de nom **gp1f4a8_default**.
- Si iniciem multiples contenidor amb l'aplicació **tonica** amb l'ajut de **docker-compose**, cada contenidor s'anomenarà **tonica-1**, **tonica-2**, **tonica-3**, **tonica-4**, etc...
- Es genera la variable d'entorn **HOST_NAME** de valor **gp1f4a8.fjeclot.net**.
- Els ports del contenidor no s'interferiran entre ells perquès són efímers.
- En el moment d'establir una connexió:
  - Connexió des de la màquina física al port 80/tcp de la màquina virtual.
  - Redirecció del **80/tcp** de la màquina host al **8000/tcp** del proxy **nginx**
  - **nginx** recull la connexió i escull un contenidor (en aquest cas per round robin)
  - **nginx** la connexió 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.

3.2- Creant el fitxer nginx.cong del proxy nginx per fer de load balancer

a) Dins de la carpeta **gp1f4a8** crea el següent arxiu de nom **nginx.conf**:

```
user  nginx;

events {
    worker_connections   1000;
}
http {
        server {
              listen 8000;
              location / {
                proxy_pass http://tonica:80;
              }
        }
}
```

b) Què fa aquesta configuració?:

- Qualsevol connexió al port **8000/tcp** del contenidor amb **nginx** és passarà al port **80/tcp** del contenidor al qual li toqui en aquell moment respondre a la petició.
- El proxy **nginx** s'encarrega a cada moment de seleccionar el contenidor que ha de respondre.
- Els serveis de xarxa que proporciona el programari docker ajudarà a nginx a treballar amb els ports efímers dels contenidors **tonica-1**, **tonica-2**, **tonica-3**, etc... per fer el redireccionament del **80/tcp** del **host** → **8000/tcp** de **nginx** → **port efimer/tcp** del **host** → port **80/tcp** intern del **docker**.

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

a) Executa dins de **gp1f4a8** l'ordre: **docker-compose up --scale tonica=5 -d**

b) Comprova que:

- S'han creat 5 contenidors de l'aplicació d'operacions matemàtiques de nom **gp1f4a8_tonica1** a **gp1f4a8_tonica5**.
- S'ha creat un contenidor amb el servidor proxy **nginx** de nom **gp1f4a8_lb_1**.
- S'ha creat una xarxa de dockers de nom **gp1f4a8_default**, executant: **docker network ls**
- Que a la xarxa **gp1f4a8_default** estan connectats **gp1f4a8_tonica1** a **gp1f4a8_tonica5** i **g12f4a8_lb_1**. Executa: **docker network inspect gp1f4a8_default | grep Name**

c) Accedeix al port **80/tcp** del servidor de la màquina **gp1f4a8** amb el naveagador utilitzant l'adreça IP **gp1f4a8** i comprova que l'aplicació funciona correctament.

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

a) Executa dins de **gp1f4a8** l'ordre: **docker-compose stop** i comprova:

- S'aturen els contenidors
- L'aplicació deixa de funcionar
- Els contenidors i la xarxa encara existeixen

b) Executa dins de **gp1f4a8** l'ordre: **docker-compose start** i comprova:

- S'inicien els contenidors
- L'aplicació torna a funcionar

c) Executa dins de **gp1f4a8** les ordres:

docker-compose stop
docker-compose down

i comprova:

- Els contenidors han desaparegut
- La xarxa ha desaparegut

3.5- Comprova el balanceig de carrega

a) Executa dins de **gp1f4a8** l'ordre: **docker-compose up --scale tonica=5** i comprova que per pantalla surten els missatges de creació i arrancada dels contenidors de l'apliacació i del proxy.

b) Accedeix amb el navegador a l'aplicació utilitzant l'adreça IP de **gp1f4a8** i comprova que l'aplicació funciona. Comprova a quins contenidors 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) De quina manera s'escull el següent contenidor al qual s'hi accedeix?. Segueix una seqüència?. Quina seqüència?. Busca el nom a internet del nom d'aquesta manera d'escollir contenidors.

e) Atura els contenidors de l'aplicació amb Ctrl+c.

f) Esborra els contenidors i la xarxa de contenidors

4- Escalant i descalant l'aplicació

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

```
docker-compose up --scale tonica=3 -d
```

Comproveu que s'executen **3** dockers de l'aplicació **tonica** amb l'ordre: **docker ps -a**

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

```
docker-compose up --scale tonica=12 -d
```

Comproveu que s'executen **12** dockers de l'aplicació **tonica** amb l'ordre: **docker ps -a**

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

```
docker-compose up --scale tonica=6 -d
```

Comproveu que s'executen 6 dockers de l'aplicació **tonica** amb l'ordre: **docker ps -a**

5- Realització d'una prova d'estrès amb siege

a) Instal·leu l'eina **siege** que permet fer proves d'estrés dins de la màquina **gp1f4a8**. Executa:

```
sudo apt-get update
sudo apt-get install siege
```

c) Inicieu l'aplicació amb docker-compose instànciant 2 contenids. Executa:

```
docker-compose up --scale tonica=2 -d
```

Comproveu que s'executen **2** dockers de l'aplicació **tonica** amb l'ordre: **docker ps -a**

b) Executa una prova d'estrès durant **60 segons** de **50 connexions d'usuaris concorrents** de la pàgina inicial de l'aplicació d'operacions matemàtiquesp. Executa des del directori gp1f4a8:

```
sudo siege -t60s -c50 http://localhost > test_stress
```

c) Obre el fitxer **test_stress** i comprova el valors Transactions, Availability, Successful transactions i Failed transacctions. A partir d'aquests valors, creus que s'haurien d'instanciar més contenidors?.

Lliurament de l'activitat

a) Mostra la imatge **llaunes:5.0** a l'equip local.
b) Mostra la imatge **llaunes:5.0** al teu compte de *DockerHub*.
c) Mostra **docker-compose.yml**
d) Mostra **nginx.conf**
e) Instancia **5** contenidors amb l'aplicació en primer terme utilitzant **docker-compose** sense l'opció **-d**
f) Accedeix a l'aplicació i comprova que per cada recarrega de l'aplicació s'utilitzen de manera consecutiva els 5 contenidors.
g) Atura l'aplicació. Inicia l'aplicació en 2n terme amb **5** contenidors l'aplicació amb l'opció **-d**.
h) Escala a **10** conenidors amb l'opció **-d**.
i) Desescala a **2** conenidors amb l'opció **-d**.
j) Mostra la prova d'estrés. Contesta la pregunta del punt **c)** de l'apartat **5**.
k) Data límit per obtenir el 100% de la nota: **dilluns 13-11-23** a les **17.00**.