

Fase 4 - Activitat 8.6: Distribució de carrega i escalabilitat.

0- Identificació del grup i activitat:

Curs: ASIX2

Projecte: PJ9 DevOps i Cloud Computing

Fase: 4

Activitat: 8.6

Grup/Individual: Individual

Membres/Alumne:

1- Introducció i objectius de l'activitat 8.6

Quan una aplicació s'ha desplegat per mitjà de contenidors Docker, 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.

A més a més, caldrà tenir un sistema per posar en marxa de manera ràpida, fàcil, segura i eficient tants contenidors com calguin per donar servei a tots els usuaris. I si disminueix la demanda, també és necessari poder disminuir els contenidors en marxa de manera ràpida, fàcil, segura i eficient de manera que alliberin recursos del sistema.

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. Un docker que treballi com a distribuïdor de carrega fa habitualment aquest feina.

L'escalabilitat és la capacitat de posar en marxa o aturar tants contenidors com calgui d'una manera ràpida, fàcil, segura i eficient en funció de la demanda que tingui l'aplicació. Utilitzant l'eina docker compose és fàcil afegir escalabilitat a una aplicació desplegada per mitjà de contenidors dockers.

Així doncs, dins d'aquesta activitat:

- Tornarem a treballar amb un xarxa de contenidors
- Afegirem un distribuïdor de carrega pel contenidors d'una aplicació.
- Utilitzarem docker compose i el fitxer docker-compose.yml per poder fàcilment desplegar un sistema que treballi amb distribució de carrega i que sigui escalable.

2- Abans de començar

a) Dins del directori **pj9** → **f4** → **a8** de la teva màquina física clona el dipòsit de **Github** que té la següent adreça URL: <https://github.com/asix2pj9/pj9f4a86.git>. Comprova que s'ha creat una carpeta de nom **pj9f4a8.6** amb la següent estructura:

- Un directori **.git** → **Eborra-ho!!!**
- Un directori **codi** amb una aplicació PHP.
- Un directori **vm** amb un fitxer **Vagrantfile**.

b) Dins del fitxer **Vagrantfile**, modifica el valor de les variables:

- **NOM_MAQUINA** de manera que **dacomo** es canviï pel teu codi format per les dues primeres lletres dels teus nom i cognoms.
- **HOSTNAME** de manera que **dacomo** es canviï pel teu codi format per les dues primeres lletres dels teus nom i cognoms.

c) Crea i posa en marxa una màquina virtual a partir del fitxer **Vagrantfile**. Accedeix a la màquina virtual i comprova que:

- El nom de sistema és correcte.
- S'ha creat la carpeta **pj9f4a86** dins de la qual, hi ha una altra de nom **codi** amb dos fitxers **.php**.

d) Dins de la màquina virtual, entra al directori **pj9f4a86**. Crea un fitxer **Dockerfile** amb el següent contingut:

```
# Imatge base a partir de la qual crearem la nostra
FROM php:8.3-apache
# Directori per defecte de treball
WORKDIR /var/www/html
# Copia la carpeta codi a "." de contenidor, a on "." és el WORKDIR
COPY codi .
# Exposa el port 80 del contenidor
EXPOSE 80
```

3- Distribució de carrega entre contenidors d'una aplicació utilitzant un proxy HTTP amb nginx.

3.1- Creant el fitxer docker-compose.yml

a) Dins de la carpeta **pj9f4a86** de la màquina virtual crea el següent arxiu de nom **docker-compose.yml**:

```
services:
  sLlauna: # Aplicació
    image: illauna:1.0
    build: .
    expose:
      - "80"
    environment:
      - HOST_NAME=pj9f4a86-dacomo.fjeclot.net
    networks:
      - xPj9f4a86

  sDc: # Distribuidor de carrega
    image: nginx:latest
    container_name: cNginx
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - sLlauna
    ports:
      - "80:8000"
    networks:
      - xPj9f4a86

networks:
  xPj9f4a86:
    name: pj9f4a86
    driver: bridge
```

NOTA 1: Canvia **dacomo** amb el teu codi format per les dues primeres lletres dels teus nom i cognoms.

NOTA 2: Les separacions i espais són molt importants quan es treballa amb fitxer de configuració format **.yml** o **.yaml**.

b) Què fa aquesta configuració?:

- Crearà un o més contenidors de nom base **pj9f4a86-sLlauna**:
 - A partir de la imatge **illauna:1.0**.
 - Exposarà el port **80/tcp** dels contenidors a un port efímer (normalment a partir de **32678/tcp**) de la màquina host (en el nostre cas, la màquina virtual).
 - Crea una variable d'entorn per poder ser utilitzada dins del contenidor de nom **HOST_NAME**.
- Crearà un contenidor de nom **cNginx**:
 - 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 **cNginx**.
 - El contenidor exporta el port **8000/tcp intern** al port **80/tcp** de la màquina host.
 - El servei de distribució de carrega **SDc** no pot iniciar-se si no s'inicia el servei **sLlauna**.
- Crearà una xarxa de contenidors de nom **pj9f4a86**.
- Està preparada per poder desplegar aplicacions amb:
 - Distribució de carrega
 - Escalabilitat

c) Aquells que tinguin interès poden llegir una aplicació sobre aquesta configuració a l'**Annex 1**.

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

a) Dins de la carpeta **pj9f4a86** de la màquina virtual crea el següent arxiu de nom **nginx.conf**:

```
user nginx;

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

Amb aquesta configuració:

- El servidor **nginx** dins **cNginx** pot fer de distribuïdor de carrega de múltiples contenidors **cLlauna**.
- El distribuïdor de carrega s'encarrega a cada moment de seleccionar el contenidor que ha de respondre a una petició d'accés l'aplicació

b) Aquells que tinguin interès poden llegir una aplicació sobre aquesta configuració a l'**Annex 2**.

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

a) Executa dins de **pj9f4a8.6** de la màquina virtual l'ordre: **docker compose up -d --scale sLlauna=5**

b) Comprova que:

- S'han creat **5** contenidors de l'aplicació d'operacions matemàtiques de nom **cllauna1** a **cllauna5**.
- S'ha creat un contenidor amb el distribuïdor de carrega **nginx** de nom **cNginx**.
- S'ha creat una xarxa de dockers de nom **pj9f4a86**
- Que tots els contenidors estan dins de la xarxa. Executa:
- Que els 5 contenidors de l'aplicació i el distribuïdor de carrega estan a la mateixa xarxa. Executa: **docker network inspect pj9f4a86 | grep Name**

3.4- Comprovació de funcionament de l'aplicació

- a) Comprova l'adreça IP de la màquina virtual.
- b) Accedeix al port **80/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.

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

- a) Executa dins de la carpeta **pj9f4a86** de la màquina virtual 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 **pj9f4a86** 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 **pj9f4a86** de la màquina virtual les ordres:

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

i comprova:

- Els contenidors han desaparegut
- La xarxa ha desaparegut
- L'aplicació torna a funcionar

3.6- Comprova el balanceig de carrega

- a) Executa dins de la carpeta **pj9f4a86** de la màquina virtual l'ordre: **docker-compose up --scale sLlauna=5** i comprova que per pantalla surten els missatges de creació i arrancada dels contenidors de l'aplicació 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 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. Deprés, esborra'ls.

4- Escalant i descalant 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 **tonica** amb l'ordre: **docker ps -a**

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

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

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

Lliurament de l'activitat

a) Mostra el nom de sistema de la màquina virtual.

b) Posa en marxa **5** contenidors amb l'aplicació càlcul del cost de llaunes de begudes. Mostra:

- Els contenidors. És important que tinguin el nom correcte.
- La xarxa de contenidors. És important que tingui el nom correcte.
- Tos els contenidors formen part de la xarxa.

c) Passa a treballar amb **7** contenidors. Mostra que funcionen visualitzant els contenidors novament.

d) Passa a treballar amb **6** contenidors que es vegin per pantalla, en primer terme.

e) Des de la màquina física accedeix a l'aplicació i comprova que funciona.

f) Des de la màquina física, refresca diverses vegades la pàgina inicial per comprovar que el balanceig de carrega funciona dins del navegador. Com ho saps que canvia de contenidor?.

g) Com saps que canvia de contenidor des del terminal?

h) Atura l'aplicació i comprova que deixa d'estar disponible.

i) Quina és la resposta a la pregunta de l'apartat **3.6.d**?

Data límit per obtenir el 100% de la nota: **dimarts 26-11-24** a les **19.10**.

ANNEX 1

Amb aquesta configuració:

- Si iniciem múltiples contenidors amb l'aplicació amb l'ajut de **docker compose**, cada contenidor s'anomenarà **pj9f4a86-sLlauna-1**, **pj9f4a86-sLlauna-2**, etc...
- Es genera la variable d'entorn **HOST_NAME** de valor **pj9f4a86-dacomo.fjeclot.net**.
- Si posem en marxa múltiples contenidors, no s'interferiran entre ells perquè 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 al port efímer de la màquina host associat al contenidor escollit.
 - El programari **docker** redirecciona el port efímer cap al port **80/tcp** intern del contenidor.

ANNEX 2

Amb aquesta configuració, si iniciem múltiples contenidors:

- 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 **pj9f4a86-sLlauna-1**, **pj9f4a86-sLlauna-2**, etc... per fer el redireccionament del **80/tcp** del **host** → **8000/tcp** de **nginx** → **port efímer/tcp** del **host** → port **80/tcp** intern del **docker**.