Informe Pràctica - DevOps

Arquitectura i Tecnologies del Software

G12_10_30

Julen Cruz Gómez 1667663

Igor Ulyanov Melnic 1667150

Part 1. Docker

Pas 1: Creació de les Xarxes Docker

Hem creat tres xarxes de tipus `bridge` (tipus de xarxa per defecte) amb uns rangs d'IPs separats per als 3 nivells de comunicació dels micro-serveis.

```
docker network create --subnet=172.18.0.0/16 --gateway=172.18.0.1
--driver=bridge FrontendNet
docker network create --subnet=172.19.0.0/16 --gateway=172.19.0.1
--driver=bridge BackendNet
docker network create --subnet=172.20.0.0/16 --gateway=172.20.0.1
--driver=bridge ManagementNet
```

Cada comanda realitza el seguent:

- Crea una xarxa virtual per a contenidors.
- Defineix un rang d'adresses IP estàtiques.
- Assigna una IP de 'gateway' per la comunicació virtual.
- Dona un nom a la xarxa

Pas 2: Creació del LoadBalancer amb NGINX

Hem creat un fitxer `nginx.conf` per configurar el balancejador, implementa una política Round-Robin per defecte. Hem especificat els servidors que ha de balancejar a la part "upstream" i la configuració del port a la part de server.

```
worker_processes auto;
events {
        worker_connections 1024;
        }
http {
        upstream frontend {
                server WebServer1:80;
                server WebServer2:80;
                server WebServer3:80;
                server WebServer4:80;
                server WebServer5:80;
        }
        server {
                listen 8000;

                location / {
                proxy_pass http://frontend;
                }
        }

}
```

Per desplegar el contenido hem descarregat una imatge oficial de Nginx del Docker Hub, i tot seguit hem executat aquesta comanda per aixecar-lo:

```
docker run -d --name LoadBalancer --hostname LoadBalancer --network
FrontendNet --ip 172.18.0.2 -p 20000:8000 -v
/mnt/c/Users/julen/Desktop/ats_2025_CC_G12/files/nginx/nginx.conf:/etc/n
ginx/nginx.conf:ro nginx
```

Aquesta comanda realitza les següents operaciones:

- Assigna nom i hostname al contenidor al network FrontendNet
- Li dona la IP 172.18.0.2
- Exposa el port '20000' per accedir al balancer des de el navegador.
- Lliga el fitxer 'nginx.conf' de l'host al contenidor (la direcció del fitxer a la màquina local està en un format especial degut a que s'ha fet la pràctica mitjançant la terminal wsl)
- Després de la direcció del fitxer a local apareix la direcció on es vol guardar el fitxer de configuración dins del contenidor.

Pas 3: Creació dels WebServers PHP + Apache

Per gestionar les peticions, hem creat una imatge personalitzada que connecta amb la base de dades i retorna la informació des d'un fitxer PHP.

El fitxer *Dockerfile* utilitzat és:

```
FROM php:8.2-apache
RUN apt-get update && apt-get install -y libpq-dev
RUN docker-php-ext-install pgsql
COPY index.php /var/www/html/index.php
```

Això, fa el següent:

- Utilitza una imatge oficial de PHP amb Apache ja integrat.
- Instal·la extensions necessàries per a la connexió amb PostgreSQL.
- Copia l'arxiu index.php dins la ruta on Apache servirà la web.

El fitxer *index.php* conté:

```
<?php
$conn = pg_connect("host=Database port=5432 dbname=AppDB user=useradmin
password=secure1234");

if (!$conn) {
    echo "<p>Error: No se pudo conectar a la base de datos.</p>";
    exit;
}

$server = gethostname();

// Insertar registro
$insertQuery = "INSERT INTO \"AppTable\" (\"WebServer\", \"Datetime\")
VALUES ('$server', CURRENT_TIMESTAMP)";
$insertResult = pg_query($conn, $insertQuery);

if (!$insertResult) {
    echo "<p>Error al insertar datos.</p>";
    exit;
}

// Consultar cantidad de veces que ha respondido
$countQuery = "SELECT COUNT(*) FROM \"AppTable\" WHERE \"WebServer\" =
'$server'";
$result = pg_query($conn, $countQuery);

if ($result) {
    $count = pg_fetch_result($result, 0, 0);
} else {
    echo "<p>Error al contar registros.</p>";
}

echo '<h1 style="font-family: Arial, sans-serif;">WebServer ID: ' .
gethostname() . ' - Num served requests: ' . $count . '</h1>';

pg_close($conn);
?>
```

Aquest fitxer index.php estableix la connexió amb la base de dades, insereix cada visita en la taula “AppTable” i mostra quantes vegades aquest servidor ha atès peticions. A més, el codi gestiona errors de connexió i consulta per garantir el correcte funcionament.

Després en situem a la carpeta on es troba el *Dockerfile* i *l’index.php*, creem la imatge *webserver-image* amb la comanda següent:

```
docker build -t webserver-image .
```

Què fa aquesta comanda?

- Crea una imatge Docker anomenada webserver-image a partir del Dockerfile.
- Inclou el fitxer index.php dins de la imatge.
- Instal·la les dependències necessàries perquè PHP pugui comunicar-se amb PostgreSQL.

Una vegada creada la imatge, despleguem els 5 WebServers amb les comandes:

```
docker run -d --name WebServer1 --hostname WebServer1 --network
FrontendNet --ip 172.18.0.11 webserver-image
docker run -d --name WebServer2 --hostname WebServer2 --network
FrontendNet --ip 172.18.0.12 webserver-image
docker run -d --name WebServer3 --hostname WebServer3 --network
FrontendNet --ip 172.18.0.13 webserver-image
docker run -d --name WebServer4 --hostname WebServer4 --network
FrontendNet --ip 172.18.0.14 webserver-image
docker run -d --name WebServer5 --hostname WebServer5 --network
FrontendNet --ip 172.18.0.15 webserver-image
```

Després i per a que cada WebServer estigui connectat tant a la network FrontendNet com a BackendNet, utilitzem aquesta comanda per a cada WebServer:

```
docker network connect BackendNet WebServerX //Cambiant X pel nombre del
WebServer
```

Pas 4: Creació de la Base de Dades PostgreSQL

Hem creat el fitxer *init.sql* amb la definició de la base de dades i la taula:

```
CREATE DATABASE "AppDB";
\c "AppDB";
CREATE TABLE "AppTable" ("WebServer" TEXT, "Datetime" TIMESTAMP);
```

A partir de la imatge oficial de PostrgeSQL, aixequem el contenidor amb la següent comanda:

```
docker run -d --name Database --hostname Database --network BackendNet
--ip 172.19.0.10 -p 15432:5432 -v
/mnt/c/Users/julen/Desktop/ats_2025_CC_G12/files/database/init.sql:/dock
er-entrypoint-initdb.d/init.sql -v pgdata:/var/lib/postgresql/data -e
POSTGRES_USER=useradmin -e POSTGRES_PASSWORD=secure1234 postgres
```

Aquesta comanda realitza les següents operaciones:

- Assigna nom i hostname al contenidor al network BackendNet
- Li dona la IP 172.19.0.2
- Exposa el port ‘15432’ per accedir al balancer des de el navegador.
- Lliga el fitxer ‘init.sql’ de l'host al contenidor (la direcció del fitxer a la màquina local està en un format especial degut a que s’ha fet la pràctica mitjançant la terminal wsl)
- Després de la direcció del fitxer a local apareix la direcció on es vol guardar el fitxer de configuración dins del contenidor.
- Crea un volum al contenidor on guardar les dades de la bbdd
- Crea un usuari ‘useradmin’ amb clau ‘secure123’

Després i per a que la Database estigui connectada també a la network ManagementNet, utilitzem aquesta comanda:

```
docker network connect ManagementNet Database
```

Pas 5: Creació de PGAdmin

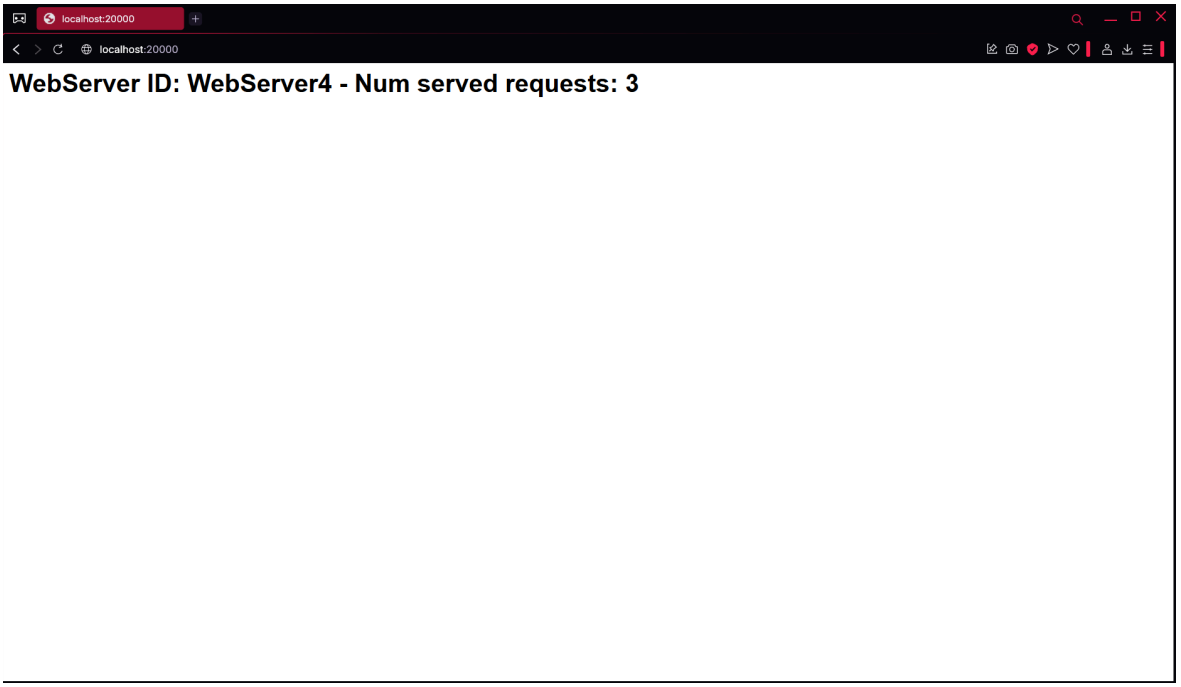
Finalment, hem desplegat un contenidor amb PGAdmin, a partir d’una imatge oficial, per gestionar la base de dades PostgreSQL des d'una interfície web:

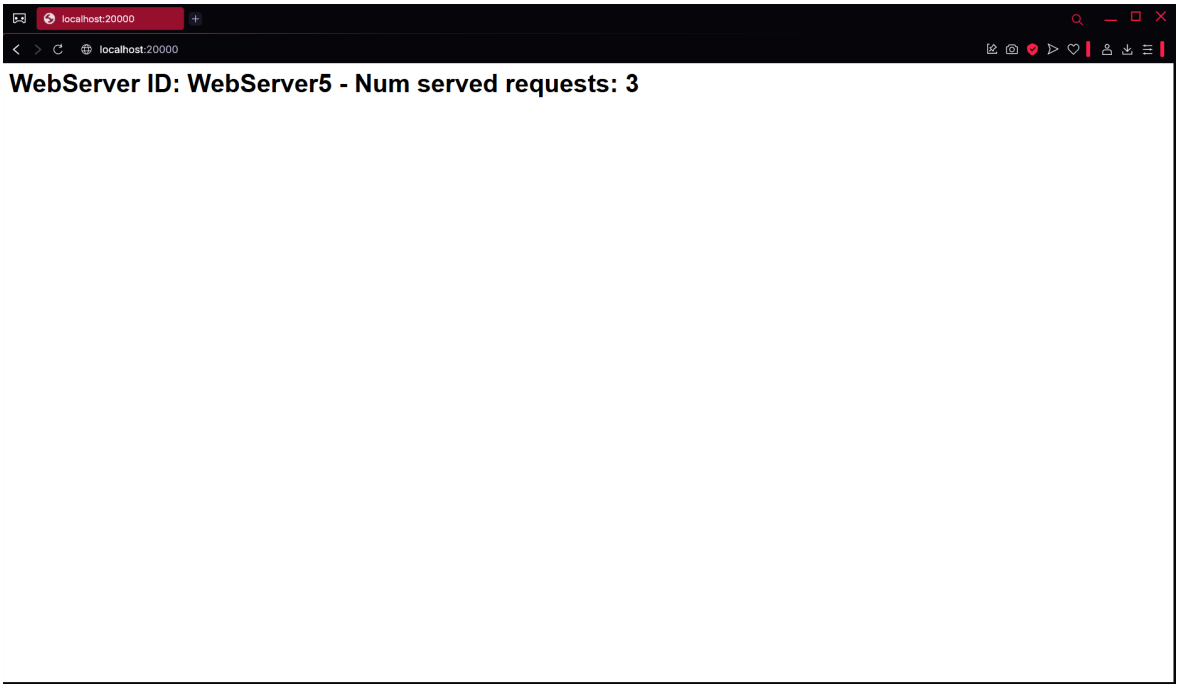
```
docker run -d --name DataBaseManager --hostname DataBaseManager
--network ManagementNet --ip 172.20.0.2 -p 20001:8000 -e
PGADMIN_DEFAULT_EMAIL=admin@admin.com -e
PGADMIN_DEFAULT_PASSWORD=secretsecret -e
PGADMIN_LISTEN_PORT=8000 -v pgadmin-data:/var/lib/pgadmin
dpage/pgadmin4
```

Aquesta comanda realitza les següents operaciones:

- Assigna nom i hostname al contenidor al network ManagementNet
- Li dona la IP 172.20.0.2
- Exposa el port ‘20001’ per accedir al balancer des de el navegador.
- Fa que el mail i clau del usuari administrador sigui ‘admin@admin’ i ‘secretsecret’ respectivament

Evidencia: Funcionament del balancejador





Evidencia: Funcionament del PGadmin

Part 2. Docker - Compose

Nuestro fichero se compone de diferentes partes, vamos a explicar los detalles que diferencian el docker compose de las instrucciones anteriormente descritas

```
services:
  loadbalancer:
    image: nginx
    container_name: LoadBalancer
    hostname: LoadBalancer
    ports:
      - "20000:8000"
    volumes:
      - ../nginx/nginx.conf:/etc/nginx/nginx.conf
    networks:
      FrontendNet:
        ipv4_address: 172.18.0.2
    depends_on:
      - webserver1
      - webserver2
      - webserver3
      - webserver4
      - webserver5
```

Primero se construyó el contenedor Load balancer. Se especifica la imagen base, nombre, hostname, puerto, volúmenes y red a la que debe estar conectado. Además se añade un parámetro “depends_on” para que este no se intente ejecutar hasta que estén subidos los contenedores WebServerX. Si no se hace esto, el contenedor LoadBalancer puede parar su ejecución inesperadamente, ya que no puede acceder a los contenedores que debe balancear.

```
  webserverX:
    build: ../php-apache
    container_name: WebServerX
    hostname: WebServerX
    networks:
      FrontendNet:
        ipv4_address: 172.18.0.11
      BackendNet:
        ipv4_address: 172.19.0.11
```

Este bloque de código aparece 5 veces en el archivo, uno para cada WebServer. Donde las “X” se sustituyen por los respectivos valores que debería tomar cada WebServer.

```yaml
database:
    image: postgres
    container_name: Database
    hostname: Database
    ports:
      - "15432:5432"
    environment:
      POSTGRES_USER: useradmin
      POSTGRES_PASSWORD: secure1234
    volumes:
      - pgdata:/var/lib/postgresql/data
      - ../database/init.sql:/docker-entrypoint-initdb.d/init.sql
    networks:
      BackendNet:
        ipv4_address: 172.19.0.10
      ManagementNet:
        ipv4_address: 172.20.0.10
```

El contenedor Database se inicializa de manera similar a como se hacía fuera del docker compose. La única diferencia es el formato del archivo y la conexión a las dos redes se hace inmediatamente, en vez de por pasos.

```yaml
databaseManager:
    image: dpage/pgadmin4
    container_name: DataBaseManager
    hostname: DataBaseManager
    ports:
      - "20001:8000"
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@admin.com
      PGADMIN_DEFAULT_PASSWORD: secretsecret
      PGADMIN_LISTEN_PORT: 8000
    volumes:
      - pgadmin-data:/var/lib/pgadmin
    networks:
      ManagementNet:
        ipv4_address: 172.20.0.2
```

El contenedor que mantiene el PGadmin se hace completamente análogo a como se hizo anteriormente. Importante destacar la línea “PGADMIN_LISTEN_PORT: 8000” para que el PGadmin escuche por defecto al puerto 8000 y no al 80.

```yaml
networks:
  FrontendNet:
    driver: bridge
    ipam:
      config:
        - subnet: 172.18.0.0/16
  BackendNet:
    driver: bridge
    ipam:
      config:
        - subnet: 172.19.0.0/16
  ManagementNet:
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.0.0/16

volumes:
  pgdata:
  pgadmin-data:
```

A continuación se especifican las 3 redes que se deben crear, con el driver por defecto y los rangos de IPs correspondientes. Y finalmente se especifican los volúmenes persistentes.