Universitat Oberta de Catalunya

Práctica

Administración de Redes y Sistemas Operativos

Grado Ingeniería Informática

16 de marzo de 2023

Índice

1 Descripción y objetivos

La práctica es una actividad individual de carácter obligatorio, que se irá elaborando durante el transcurso del semestre y que deberá estar finalizada cuando se realice la entrega final según el calendario de la asignatura.

El objetivo de la actividad es poner en práctica diversos conocimientos teóricos adquiridos durante el transcurso del curso mediante la lectura de los diferentes módulos y fomentar la investigación de nuevos conceptos técnicos que complemente lo aprendido. De esta forma se pretende que el estudiante adquiera algunas de las competencias asociadas a la administración de redes:

- Utilizar varios servicios o programas que muestran diferentes modos de intercambiar información.
- Utilizar las herramientas administrativas y de análisis de red más comunes.
- Implementar mecanismos de filtrado de paquetes.
- Experimentar los conceptos básicos de la comunicación, incluyendo arquitecturas, protocolos y servicios.

Además de la adquisición de competencias más transversales.

- Capacidad de análisis y de búsqueda de información que complementen los conocimientos obtenidos para solucionar problemas complejos
- Mejora en la capacidad de redacción y presentación de informes técnicos

2 Enunciado

Desde el departamento de desarrollo de una empresa se nos solicita poder disponer de una infraestructura donde alojar su servicio web. Como requisito principal se indica que ésta ha de poder ser accesible mediante un navegador o por comandos de terminal desde el exterior y hace uso de una base de datos la qual no ha de ser accesible desde internet. Como medida de seguridad, el servidor tan solo debe exponer los puertos 80 (HTTP) y 443 (HTTPS) siendo éste último el protocolo final de navegación.

2.1 Desarrollo práctico

Para ofrecer este tipo de servicio existen varias arquitecturas: basadas en servicios alojados en una sola máquina, distribuidas, virtualizadas y/o basadas en contenedores. Para facilitar su desarrollo se proporciona la infraestructura a implementar. En este caso, se ha optado por diseñar el servicio basado en técnicas de virtualización y contenedores tal como se muestra en la figura 1

Básicamente se tratará de realizar la implementación de los siguientes sistemas:

1. Máquina virtual de alojamiento de los servicios [1].
2. Implementación del servidor de base de datos [1, 2].
3. Implementación del servidor web [1, 3].

4. Implementación del servidor *reverse-proxy* [1, 3].
5. Diseño y configuración del cortafuegos.[4, 5]

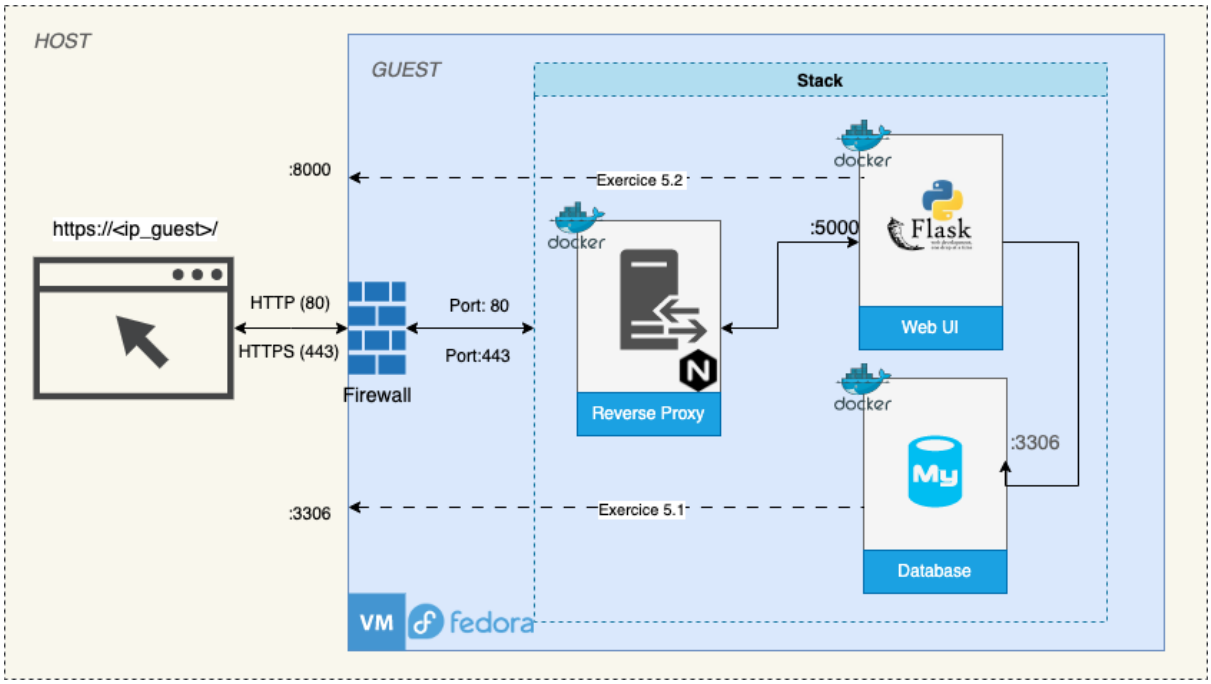

Figura 1: Diagrama de la infraestructura a implementar

Para realizar la práctica será necesario el uso de una máquina **virtual** como hospedaje de los servicios, utilizando el software de virtualización que elijáis entre los diversos que hay disponibles (VirtualBox, VmWare, Azure,etc..). Posteriormente, se irán implementando los diferentes sistemas mencionados anteriormente en forma de ejercicios que formarán parte de la composición final de la infraestructura. Todo el proceso se documentará con detalle en un informe que será el documento que finalmente se tendrá que librar (ver apartado 8 de forma de entrega).

3 Preparación del entorno

- Escoger un software de **virtualización** e instalarlo en vuestro ordenador si fuése necesario.
- Descargar la imagen ISO del sistema operativo en vuestro ordenador. El sistema operativo de este servidor virtualizado será:

  Fedora 37 server (https://getfedora.org/es/server/download/).
- Adicionalmente, se necesitará la herramienta ***nmap*** (https://nmap.org) de rastreo de puertos y la herramienta de transferencia de datos ***curl*** (https://curl.se) tanto en vuestro sistema *Host* o sistema externo como en la máquina virtual, si no las incorpora el propio sistema operativo.
- En la máquina virtual, deberéis disponer de la herramienta ***git*** (https://git-scm.com/) de control de versiones y la herramienta de contenedores

  Docker engine (https://docs.docker.com/engine/install/) con el complemento *docker-compose-plugin*

4 Ejercicio 1: Instalación del S.O. en la máquina virtual (0,5 puntos)

El primer paso a realizar será la configuración de la máquina virtual y la instalación del sistema operativo indicado anteriormente. Para ello:

1. Configurar la red del sistema de virtualización con una o dos NIC que permita conectividad con Internet y conectividad entre vuestro PC (Host) y la máquina virtual (Guest) [1].
2. Configurar la máquina virtual con dos discos virtuales e instalar el sistema operativo teniendo en cuenta que se desea configurar los dos discos virtuales en una formación **RAID 0 o RAID 1** (a escoger) para obtener una capacidad total de **10 GB**
3. El nombre de la máquina deberá ser **ARSO20222** y el **usuario** con permisos de administración a crear sea el mismo que vuestro **usuario del campus**. Este usuario será el que se utilice para realizar todos los comandos [2].

4.1 Evidencias

Una vez finalizada la instalación comprobar y adjuntar en el informe:

- Razonamiento de la elección del RAID configurado para obtener la capacidad deseada, y captura de pantalla de la configuración de los discos durante la instalación (**no es necesario adjuntar capturas de toda la instalación**).
- Pantalla del terminal que muestre el tamaño y estado del Raid con las instrucciones:

```
$ df -h
$ cat /proc/mdstat
```

- Verificación de la conexión entre el *Host* i el *Guest*

```
$ ping -c1 <ip_guest>
```

- Anotar en la cabecera de las páginas del informe las dos IPs: IP_HOST, IP_GUEST para tener clara dichas referencias durante el informe.

[1] Ejemplo de configuraciones: https://www.virtualbox.org/manual/ch06.html#networkingmodes
[2] Se recomienda realizar dichas configuraciones durante la instalación

5 Ejercicio 2: Stack de servicios (4 puntos)

Configuración de un stack de servicios con `Docker compose`

La idea de este apartado es poner en marcha los servicios correspondientes a la web a publicar (*Mysql* [6] y servidor web con *Python/Flask* [7]) de tal forma que interactúen entre ellos. Para hacerlo posible es necesario utilizar la herramienta `Docker compose` que nos permite definir y compartir aplicaciones de diversos contenidos mediante la definición en un archivo de configuración llamado `docker-compose.yml` [8]

- Accede al sistema virtualizado *Guest*
- Actualiza los paquetes. Instala *Docker engine* con el complemento *docker-compose-plugin* en el sistema
- Descarga la estructura básica de ficheros que va a servir como base para realizar los ejercicios mediante la instrucción:

```
$ git clone https://github.com/jestebangr/prac20222-orig.git
```

5.1 Servidor *Mysql*

Se empezará el *Stack* de servicios configurando el servicio *Mysql*.

El archivo `docker-compose.yml` proporcionado debe tener un aspecto como el que se muestra a continuación:

```
version: '3.9'
services:
  mysql:
    image: 'mysql:5.7'
    container_name: mysql
```

Se utilizará la imagen oficial *Mysql* de *dockerhub* etiquetada como ***mysql:5.7***

Para poner en marcha *Stack* habrá que situarse en la carpeta del proyecto *git* y ejecutar el comando: (–build para reconstruir la imagen, -d si lo desea levantar en *background*) :

```
$ sudo docker compose up --build
```

Para detener el *Stack* iniciado con la opción `-d` se puede usar el comando:

```
$ sudo docker compose down
```

Acabar de configurar el servicio teniendo en cuenta que:

- Se debe exponer el puerto de ***Mysql*** (**3306**) para poder hacer pruebas de acceso desde la máquina *Guest*.

- Se necesitará crear un **volumen** para que los datos de *Mysql* sean persistentes y no se borren cuando reiniciemos el contenedor.
- Al iniciar el servicio, hay que asegurarse de:
  - proporcionar un acceso para vuestro usuario y una contraseña cualquiera.
  - crear una **BBDD** de nombre **axso-db**
  - cargar los datos de la tabla **usuario** que encontraréis en el fichero `data.sql` del directorio `db`

Se puede consultar la documentación oficial de la imagen de *Docker Mysql* [6] que se utiliza para encontrar cómo pasar toda la información requerida al contenedor

Se valorará el hecho de no poner información sensible directamente dentro del archivo `docker-compose.yml`. Para ello se pueden utilizar variables de entorno [9]

5.1.1 Evidencias

- Contenido del archivo `docker-compose.yml` completado.

```
$ cat docker-compose.yml
```

- Contenido de otros archivos modificados y puesta en marcha del *Stack*.
- captura de pantalla de los contenedores en ejecución:

```
$ sudo docker ps
```

- Comprobación del estado del puerto *Mysql*:

  Desde la máquina Guest:

```
$ sudo netstat -a | grep mysql
```

  Desde el PC (cualquiera de las opciones):

```
$ nmap -p- --open --min-rate=5000 -Pn -v -sS -n <ip_guest>
$ nc -z4nv <ip_guest> <puerto>
```

- Una vez levantado el servicio satisfactoriamente, intentar y mostrar el acceso al *Mysql* con un cliente cualquiera. Utilizar el usuario con las credenciales que se han configurado en el contenedor.

5.2 Servidor web Python/Flask

Una vez configurado el *Mysql* crear otro servicio dentro del *Stack* que proporciona el servidor web capaz de entender el lenguaje Python en servidor.

Se usará la imagen oficial de Python con alpine de *Dockerhub*[10] etiquetada como ***python:3.11.2-alpine***.

El fichero de trabajo `docker-compose.yml` tendrá que tener ahora un aspecto similar a:

```
version: '3.9'
services:
  mysql:
    image: 'mysql:5.7'
    container_name: mysql
    (...)
  web:
    build: ./web
    container_name: web
    (...)
```

Completar la configuración del servicio `web` en el archivo `Dockerfile` incluido en el directorio `web/`

- Hay que conseguir que el contenedor disponga de los archivos necesarios para levantar el servidor web con python. Estos archivos los podéis encontrar en el directorio `app/` que se os ha proporcionado. Se tienen que alojar en el directorio `/usr/src/app` del contenedor. Tenéis que añadir las instrucciones necesarias al fichero `Dockerfile` para que el contenedor disponga de estos archivos. En este apartado no se permiten usar volúmenes. [7]
- No debéis modificar ningún archivo de la carpeta `app/`.
- Recordad que cualquier cambio que hagáis en el fichero `Dockerfile`, deberéis reconstruir el contenedor con el comando:

  ```
  $ sudo docker compose build
  ```

Completar la configuración del servicio `web` en el archivo `docker-compose.yml` teniendo en cuenta que:

- Se debe **exponer el puerto 8000** para poder hacer pruebas de acceso desde un navegador del *Host*.
- Una vez esté todo bien configurado, podréis levantar el *Stack* con el comando:

  ```
  $ sudo docker compose up
  ```

Podéis consultar la documentación oficial de la imagen de *Docker Python/Flask* [7]

Ayudaros también de la referencia del fichero `Dockerfile` [11]

5.2.1 Evidencias

- Contenido del fichero `docker-compose.yml` completado.

```
$ cat docker-compose.yml
```

- Contenido del fichero `Dockerfile` completado

```
$ cat Dockerfile
```

- Levantar el *Stack* de servicios tal como se ha indicado en el apartado 5.1 y comprobar y mostrar si el contenedor ***web*** se ha iniciado satisfactoriamente.

```
$ docker ps
```

- Probar a acceder desde un navegador del *Host* a la url http://localhost:8000/ donde muestre la página web con mensaje satisfactorio y pendiente de conexión a la base de datos. Adjuntar una captura de pantalla del navegador.

5.3 Servidor web Python/Flask con Mysql

A continuación se tendrá que enlazar el contenedor ***web*** con el contenedor ***mysql***. Hay que hacer que el *web* sea capaz de conectarse a la BBDD ***axso-db***.

Aspectos a tener en cuenta:

- Hay que pasarle al contenedor ***web*** las variables de entorno adecuadas para que el servidor Python/Flask pueda usarlas e intente realizar la conexión.
- Podéis ayudaros examinando el fichero *web/app/app.py* para consultar qué variables de entorno son necesarias para conectarse a la BBDD ***axso-db***
- Recordad que no podéis tocar ningún fichero del directorio */app*

5.3.1 Evidencias

- Contenido de los ficheros modificados indicando los cambios realizados.
- Captura de pantalla de los contenedores en ejecución:

```
$ sudo docker ps
```

- Comprobación que se están exponiendo los puertos 8000 y 3306:

  Desde la máquina Guest:

```
$ sudo netstat -tulpn | grep LISTEN
```

Desde el PC (cualquiera de las opciones):

```
$ nmap -p- --open --min-rate=5000 -Pn -v -sS -n <ip_guest>
$ nc -z4nv <ip_guest> <puerto>
```

- Probar a acceder desde un navegador del *Host* a la url `http://<ip_guest>:8000/` donde muestre la página web con mensaje satisfactorio y el contenido de la base de datos. Adicionalmente, comprobar que la API de la base de datos es correcta en la url `http://<ip_guest>:8000/db`

  Adjuntar una captura de pantalla de lo que se muestra en el navegador.

6 Ejercicio 3: *reverse-proxy* (3 puntos)

Con el fin de proteger las identidades de los servidores internos y como defensa adicional contra ataques de seguridad, se aconseja instalar y configurar un servidor *reverse-proxy* tal y como se indica en el diagrama 1 (pág.3) de forma que muestre el contenido del servidor web del ejercicio 5.3 (pág.8). Se debe tener en cuenta lo siguiente:

- El servidor proxy deberá crearse usando un servidor ***nginx*** dockerizado [12].
- El contenedor Docker deberá llamarse: **reverse-proxy**
- Deberá aceptar peticiones HTTP y HTTPS a través de los puertos 80 y 443 respectivamente.
- La web debe ser accesible del `Host` (PC) a través de la URL:
  https://<ip_guest>/

Para facilitar la tarea, se realizará en dos fases: primero funcionando con el protocolo HTTP y después añadiendo el protocolo seguro HTTPS.

6.1 Proxy por HTTP

Crear y configurar según sea necesario, tres archivos: `Dockerfile`, `nginx.conf` [13] y `docker-compose.yml`. Incluir los archivos en el directorio `reverse-proxy`.

Hay que tener en cuenta que el archivo de configuración `nginx.conf` que se genera en local es necesario traspasarlo al contenedor en tiempo de creación (`build`) del contenedor. Este archivo de configuración debería tener un formato del estilo:

```
server {
 /* Configuración por puerto 80 y proxy en web interna */
}
```

Cualquier modificación sobre este archivo, representará tener que ejecutar:

```
$ sudo docker compose up --build
```

Hay que tener en cuenta también que, para que el *reverse-proxy* pueda acceder al contenedor *web-flask*, deben estar en la misma red.

- Configurar el `docker-compose.yml` creado y el de *Stack* para comunicar ambos contenedores a través de una red docker [14] llamada ***web-network*** y vigilar cuál será ahora el nombre y puerto correcto de redirección de *reverse-proxy*.
- Modificar también el `docker-compose.yml` de *Stack* para conectar de forma adicional el contenedor *web-php* y *mysql* a través de otra red llamada ***db-network***

`docker-compose.yml` debería tener un contenido del tipo:

```
services:
    (...)
networks:
  web-network:
    name: web-network
```

6.1.1 Evidencias

- Contenido desde el terminal de los archivos modificados:

```
$ cat Dockerfile
$ cat nginx.conf
$ cat docker-compose.yml
```

- Comprobación de que el servidor *Guest* tiene el puerto 80 abierto y que son accesibles desde *Host*.

  Desde la máquina *Guest*:

```
$ sudo netstat -a | grep http
```

  Desde el PC (cualquiera de las opciones):

```
$ nmap -p- --open --min-rate=5000 -Pn -v -sS -n <ip_guest>
$ nc -z4nv <ip_guest> 80
```

- Información de los contenedores en cada red:

```
$ sudo docker network inspect -f '{{json .Containers}}'
 ↪  web-network | python3 -m json.tool
$ sudo docker network inspect -f '{{json .Containers}}'
 ↪  db-network | python3 -m json.tool
```

6.2 Proxy por HTTPS

Modificar la configuración del servidor *proxy* para que todas las peticiones HTTP se reenvíen automáticamente al protocolo seguro HTTPS.

Nota: Para poder utilizar el protocolo HTTPS deberá generar certificados autofirmados con el comando `openssl` para obtener dos archivos: ***.crt** y ***.key**. En la dirección de correo del certificado incluir vuestro email de la UOC.

Habrá que modificar la configuración `nginx.conf` para tener en cuenta el protocolo 443 y la redirección y habrá que volver a generar el contenedor (`build`). Este fichero de configuración debería tener ahora un formato del estilo:

```
server {
 /* Config. por puerto 80 con modificaciones de redirección a https */
}
server {
 /* Config. por puerto 443 con certificados y proxy en la web interna*/
}
```

6.2.1 Evidencias

- Contenido desde el terminal de los archivos `Dockerfile` y `nginx.conf`:

  ```
  $ cat Dockerfile
  $ cat nginx.conf
  ```

- captura de pantalla de los contenedores en ejecución:

  ```
  $ sudo docker ps
  ```

- Comprobación de que el servidor *Guest* tiene los puertos 80 y 443 abiertos y que son accesibles desde *Host*.

  Desde la máquina Guest:

  ```
  $ sudo netstat -a | grep http
  ```

  Desde el PC (cualquiera de las opciones):

  ```
  $ nmap -p- --open --min-rate=5000 -Pn -v -sS -n <ip_guest>
  $ nc -z4nv <ip_guest> <puerto>
  ```

- Probar a acceder desde un navegador del *Host* a la URL:

  http://<ip_guest>/

  Adjuntar una captura de pantalla de lo que se muestra en el navegador.

- Pantalla del navegador del cliente *Host* con la consola de inspección web abierta donde se muestre el aviso 301 con la redirección o ejecución del comando en `Host`:

  ```
  $ curl -I http://<ip_guest>/
  ```

- Pantalla del navegador del cliente *Host* con los **detalles del certificado**

7 Ejercicio 4: Cortafuegos (0,5 puntos)

Por último, para proteger el servidor *Guest* de accesos a otros servicios no necesarios, se pide:

- Elegir un cortafuegos y configurarlo para que sólo se pueda acceder al servidor *Guest* desde el exterior mediante los puertos **80** y **443**.
- Adicionalmente, a modo experimental, configurarlo para que no se pueda acceder desde el sistema *Guest* a las siguientes URLs:

  https://www.twich.tv/ y https://www.uoc.edu/

7.1 Evidencias

- Listado, desde el terminal, de las reglas activas aplicadas al *firewall*.
- Comprobación desde el *Host* de los puertos abiertos en *Guest*:

```
$ sudo nmap -p- --open --min-rate=5000 -Pn -v -sS -n <ip_guest>
$ nc -z4nv <ip_guest> <puerto>
```

  Si se usa el comando `nc`, comprobar que los puertos expuestos en el ejercicio 2 ya no son accesibles.

- Salida de los comandos desde *Guest* para comprobar URLs bloqueadas:

```
$ curl -sSL -D - <url> -o /dev/null
```

- ó Captura de pantalla de un navegador del sistema *Guest* con el error obtenido al intentar acceder a estas URLs bloqueadas.
- ¿Podríamos cerrar el resto de puertos expuestos por el *Stack* de servicios que no sean el 80 y 443 de alguna manera más aparte de utilizar el cortafuegos? Razona tu respuesta y muestra los cambios de configuración en caso necesario.

8 Entrega (fondo y forma: 2 puntos)

La práctica consiste en la entrega de un informe (un documento escrito en formato PDF), que contenga lo siguiente:

- Portada e índice de contenido y figuras
- Un capítulo por cada ejercicio done se muestre cada una de las evidencias respectivas
  - Las capturas de terminal deben ser claras y legibles
  - Los párrafos de texto que hagan referencia a comandos y/o contenido de archivo deberán ser con una fuente proporcional y diferente al resto del texto.
- Un apartado final con las conclusiones que haya extraído de la elaboración de la práctica.
- Un anexo con el contenido de los archivos de configuración que se hayan creado/modificado.
- La práctica deberá entregarse a más tardar en la fecha máxima de entrega definida en el calendario de la asignatura.

9 Evaluación

- La práctica es una actividad individual de carácter obligatorio que constituye el 30 % de la evaluación global de la asignatura.
- El plagio total o parcial detectado a la práctica supondrá una calificación de 0.
- La nota de la práctica tendrá que ser como mínimo de 5 sobre 10 para poder superar el global de la asignatura. Una nota inferior o no presentar la práctica en los plazos establecidos representará el suspenso del global de la asignatura.
- El contenido del informe que no contenga las descripciones, explicaciones y razonamientos necesarios no se puntuará.
- La solución de los diferentes apartados no es única, en la evaluación de la corrección de la solución dada se tendrá en cuenta aspectos como eficiencia, elegancia, simplicidad.

Referencias

[1] Eduardo Marco Galindo y Javier Panadero Martínez. *Administració de servidors / Administración de servidores*. Ed. por UOC. PID_00275601. UOC, 2022.

[2] Manel Mendoza Flores, Miquel Colobran Huguet y Javier Panadero Martínez. *Administració de les dades / Administración de los datos*. Ed. por UOC. PID_00275599. PID_00275599. UOC, 2022.

[3] Alberto José Mateos Bartolomé, Joan Ramon Esteban Grifoll y Javier Panadero Martínez. *Administració dels serveis web /Administración de servicios web*. Ed. por UOC. PID_00275605. PID_00275605. UOC, 2022.

[4] Miguel Martín Mateo et al. *Administració de xarxa / Administración de redes*. PID_00275602. UOC, 2022.

[5] José Manuel Castillo Pedrosa y Javier Panadero Martínez. *Administració de la seguretat / Administración de la seguridad*. Ed. por UOC. PID_00275600. PID_00275600. UOC, 2022.

[6] Dockerhub. *Docker Mysql Documentation*. 2023. URL: https://hub.docker.com/_/mysql.

[7] Dockerhub. *Docker Python Documentation*. 2023. URL: https://hub.docker.com/_/python.

[8] Docker Documentation. *Compose file version 3 reference*. Feb. de 2022. URL: https://docs.docker.com/compose/compose-file/.

[9] Docker Documentation. *Environment variables in compose*. Feb. de 2022. URL: https://docs.docker.com/compose/environment-variables/.

[10] Dockerhub. *Dockerhub Documentation*. 2022. URL: https://hub.docker.com.

[11] Docker Documentation. *Dockerfile reference*. Feb. de 2022. URL: https://docs.docker.com/engine/reference/builder/.

[12] Docker Nginx. *Docker Hub*. 2022. URL: https://hub.docker.com/_/nginx.

[13] Nginx Reverse proxy. *Nginx Reverse Proxy*. 2022. URL: https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/.

[14] Docker Network. *Networking in compose*. 2022. URL: https://docs.docker.com/compose/networking/.