Docker

En esta práctica daremos los primeros pasos en Docker y observaremos sus principales características.

9.1. Instalación

Como indica en https://docs.docker.com/engine/install/ para la instalación hay muchas variantes. En Windows https://docs.docker.com/docker-for-windows/install/

En Ubuntu:

```
sudo apt install docker.io
sudo groupadd docker
sudo usermod -aG docker $USER
```

Es necesario como mínimo salir de la sesión y volver a entrar para que la cuenta usuaria actualice los grupos. Se comprueban con la instrucción `id`.

En las salas de ordenadores de la escuela ya está instalado, en la cuenta `udocker`.

9.2. Manejo básico

Para arrancar un microservicio (en este caso sólo es un comando de ubuntu para comprobar que se usa el mismo kérnel y borra con `-rm` el contenedor):

```
docker pull ubuntu
docker images # imágenes, tenemos el ubuntu, observad el tamaño
docker run --rm ubuntu uname -a # ejecuta el comando

docker run --rm -it ubuntu bash # interactivo, línea de comandos
  sleep 100 & # en segundo plano
  ps aux # vemos su PID, el del bash que arranca el contenedor y el ps aux
# desde fuera del contenedor, otra línea de comandos
ps aux | grep sleep # vemos que es el único sleep con otro PID

docker ps -a # contenedores en ejecución y parados con el -a, se han
   borrado
docker run ubuntu uname -a # no borra el contenedor
docker ps -a # contenedores en ejecución y parados con el -a
docker rm practical_lehmann # borra el contenedor usando el nombre de NAMES
docker images # sigue estando la imagen
docker rmi ubuntu # borra la imagen
```

Fichero `Dockerfile` mínimo para crear una imagen:

```
FROM scratch
COPY busybox /
```

Fichero 9.1: `Dockerfile`

Crear la imagen con

```
cp /bin/busybox .
docker build --tag ejemplo .
```

Arrancar para hacer un `ls -l` dentro del contenedor con

```
docker run --rm ejemplo ./busybox ls -l
```

y para crear una línea de comandos interna:

```
docker run -it --rm --name "EjemploDocker" -h "ED" ejemplo ./busybox ash
# y dentro por ejemplo:
uname -a
exit  # para salir
docker run -it --rm --name "Ejdocker" -h "ED" ejemplo ./busybox pwd
/ # devuelve pwd que es la raíz de su sistema de ficheros del contenedor
```

9.3. Servidor web

Para arrancar un microservicio con un solo servicio servidor web apache. Lo arranca como demonio (en segundo plano), con nombre *web* y con puerto externo 8080 e interno 80, imagen httpd y un volumen *sitio* para ese directorio que contiene la web.

```
docker run -dit --name web -p 8080:80 -v sitio:/usr/local/apache2/htdocs/
   httpd
docker ps
docker volume ls
docker volume inspect sitio
sudo ls -l /var/lib/docker/volumes/sitio/_data
sudo vi /var/lib/docker/volumes/sitio/_data/index.html
```

Para ver que funciona accedemos `http://localhost:8080/` y para comprobar que la modificación funciona.

También podemos entrar dentro del contenedor para editar.

```
docker ps -a
docker start web # si ha parado
docker exec -it web bash # un nuevo proceso dentro del contenedor
apt update
apt install vim
cd htdocs
vim index.html
```

Y podemos ver el cambio en http://localhost:8080/. Luego paramos y borramos el contenedor, y cuando arrancamos otro se siguen viendo los cambios.

```
docker stop web
docker rm web
docker ps -a
docker run -dit --name web -p 8080:80 -v sitio:/usr/local/apache2/htdocs/
   httpd
```

9.4. docker-compose

Instalación de docker-compose en /usr/local/bin o en un fichero local por si no tenemos permisos de root como en las aulas:

```
sudo curl -L  "https://github.com/docker/compose/releases/download/1.25.5/
   docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
curl -L  "https://github.com/docker/compose/releases/download/1.25.5/docker
   -compose-$(uname -s)-$(uname -m)" -o docker-compose # fichero local,
   ejecutar con ./docker-compose
alias docker-compose=./docker-compose # para usarlo como si estuviera en el
    PATH
```

Por apt

```
sudo apt install docker-compose
```

Para arrancar el grupo de contenedores Docker especificados en el docker-compose.yml que levanta el servicio Nextcloud se hace:

```
cd nextcloud/
docker-compose up -d
docker ps -a
docker-compose logs # evolución, registros
```

En http://localhost:88/ se abre la configuración. Hay que poner nombre y contraseña (nuevos) y los datos del YML: base de datos MariaDB, usuario *nextcloud*, *Contraseña*, nombre de la BD *nextcloud* y host de la BD *db:3306* (es el puerto por defecto de MariaDB).

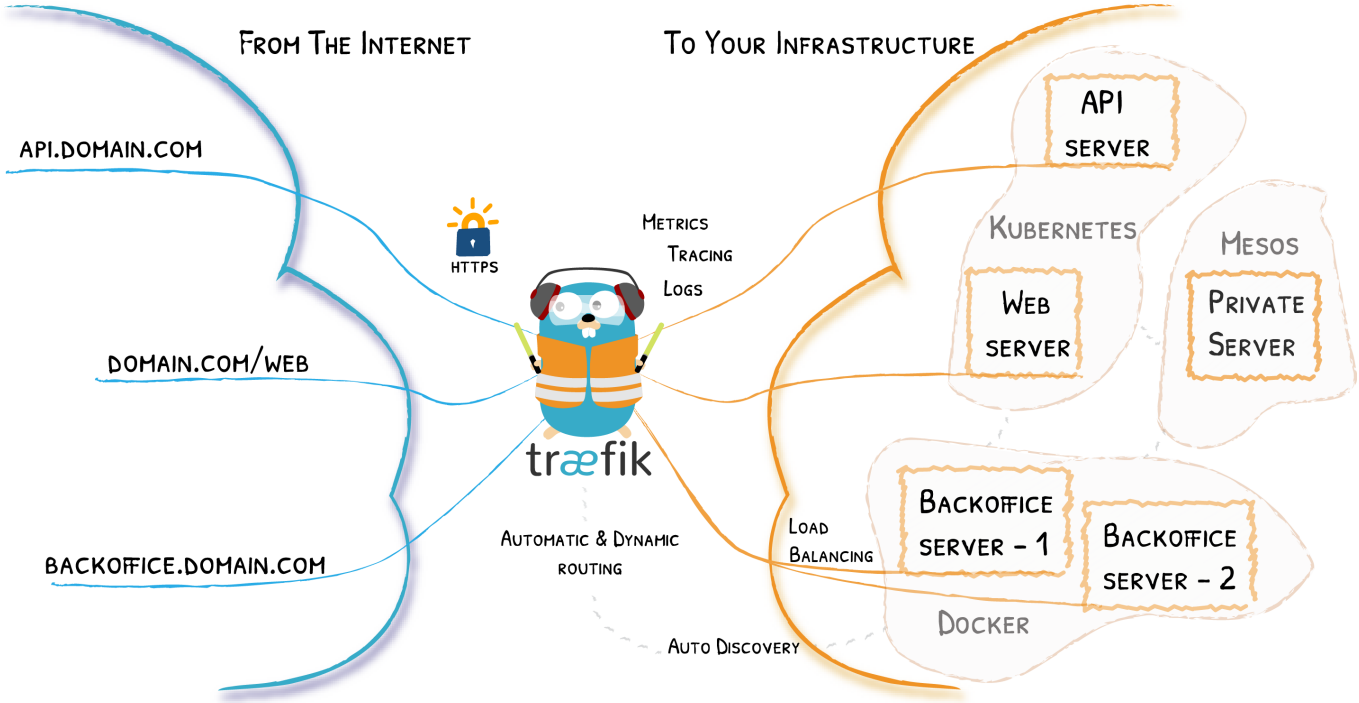
Para parar:

```
docker-compose down # borra contenedores pero no datos
docker volume ls    # listado de volúmenes
docker volume prune # borra los volúmenes no usados
```

Si no borramos los volúmenes podemos volver a arrancar con los mismos datos y con nuevos contenedores, actualizando versiones si se especifica la etiqueta o *tag* latest u otra en el YML.

9.5. *Træfik* como proxy inverso

En esta parte observaremos las principales características de Træfik.



Vamos a lanzar varios servicios que serán accesibles a través de *Træfik*, un proxy inverso que crea los caminos de las webs de forma automática y dinámica, definidos con etiquetas en otros ficheros `docker-compose`.

Para arrancar, entramos en el directorio `traefik`:

```
docker network create traefik
SERV='localhost' docker-compose -f traefik.yml up -d --scale whoami=2
SERV='localhost' docker-compose -f docker-compose-wp.yml up -d
SERV='localhost' docker-compose -f docker-compose-nextcloud.yml up -d
curl -H Host:nextcloud.localhost http://nextcloud.localhost/ # si no
   accedemos por web
```

El fichero `traefik.yml` arranca el propio contenedor `traefik`, dos instancias de `whoami` que hacen balanceo de carga (responden alternativamente) con `-scale whoami=2`, y `Portainer`, un gestor gráfico de los contenedores.

El panel de control de *Træfik* es accesible en `http://localhost/traefik/`, el de *Portainer* en `http://portainer.localhost/` y las *whoami* en `http://localhost/whoami`

El fichero `traefik.yml` en la siguiente página:

```
# SERV='localhost' docker-compose -f traefik.yml up -d --scale whoami=2
# ver https://lsi.vc.ehu.eus/pablogn/docencia/ISO/Act9%20Docker/

version: '3.3'

services:
  traefik:
    image: traefik:v1.7-alpine
    container_name: "traefik"
    command: --api --docker --docker.domain=${SERV} --logLevel=DEBUG
    ports:
      - "80:80"
      - "8080:8080"
    labels:
      - "traefik.docker.network=frontend"
      - "traefik.enable=true"
      - "traefik.frontend.rule=Host:${SERV}; PathPrefixStrip:/traefik"
      - "traefik.port=8080"
      - "traefik.protocol=http"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - /dev/null:/traefik.toml
    networks:
      - "traefik"

  whoami:
    image: containous/whoami
    labels:
      - "traefik.enable=true"
      - "traefik.backend=whoami"
      - "traefik.frontend.rule=PathPrefixStrip: /whoami"
      - "traefik.http.routers.whoami.rule=Host(${SERV})"
      - "traefik.http.routers.whoami.entrypoints=web"
    networks:
      - "traefik"

  portainer: # https://www.smarthomebeginner.com/traefik-2-docker-tutorial
     /#Portainer_with_Traefik_2_and_OAuth
    image: portainer/portainer
    container_name: portainer
    restart: unless-stopped
    command: -H unix:///var/run/docker.sock
    networks:
      - traefik
      - default
    labels:
      - "traefik.enable=true"
      - "traefik.protocol=http"
      - "traefik.docker.network=frontend"
```

```
      - "traefik.frontend.rule=Host:portainer.${SERV}"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data

    security_opt:
      - no-new-privileges:true

volumes:
  portainer_data:

networks:
  traefik:
    external:
      name: traefik
```

Fichero 9.2: `traefik.yml`

El `docker-compose-wp.yml` lanza un contenedor Wordpress (o más réplicas) apoyado en un contenedor que gestiona la base de datos MySQL en la red `wp_back` y accesible en el camino `http://wp.localhost/`

```
# docker-compose up -d --scale wordpress=3 --scale db=2

version: '3'

services:
   wpdb:
     image: mysql:5.7
     volumes:
       - wpdb:/var/lib/mysql
     restart: unless-stopped
     labels:
       - traefik.enable=false
     networks:
       - wp_back
     environment:
       MYSQL_ROOT_PASSWORD: wordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - wpdb
     image: wordpress:latest
     labels:
      - "traefik.enable=true"
      - "traefik.port=80"
      - "traefik.backend=wordpress"
      - "traefik.frontend.rule=Host:wp.${SERV}"
  #    - "traefik.frontend.rule=Host:${SERV}; PathPrefix:/wp"
  #    - "traefik.frontend.rule=PathPrefixStrip:/wp"
      - "traefik.docker.network=traefik"
     restart: unless-stopped
 #    ports:
 #      - "80:80"
     networks:
      - traefik
      - wp_back
     environment:
       WORDPRESS_DB_HOST: wpdb:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
     volumes:
       - wp:/var/www/html
     deploy:
       mode: replicated
```

```
      replicas: 2
      restart_policy:
        condition: always
      labels:
        APP: WORDPRESS

volumes:
    wpdb:
    wp:

networks:
  wp_back:
    driver: bridge
  traefik:
    external:
      name: traefik

# docker-compose scale wordpress=3
```

Fichero 9.3: `docker-compose-wp.yml`

El `docker-compose-nextcloud.yml` lanza un contenedor con la nube privada, apoyado en un contenedor que gestiona otra base de datos MariaDB (que también usa el mismo puerto pero está en una red distinta `nc_back`) y accesible en el camino `http://nextcloud.localhost/`

```
### https://github.com/nextcloud/docker
## https://blog.ssdnodes.com/blog/installing-nextcloud-docker/
# https://docs.nextcloud.com/server/16/admin_manual/configuration_server/
   config_sample_php_parameters.html

# SERV='localhost' docker-compose -f docker-compose-nextcloud.yml up -d
# SERV='localhost' docker-compose -f docker-compose-nextcloud.yml ps
# SERV='localhost' docker-compose -f docker-compose-nextcloud.yml logs
   nextcloud
# SERV='localhost' docker-compose -f docker-compose-nextcloud.yml exec
   nextcloud bash

version: '2'

volumes:
  nextcloud:
  ncdb:

services:
  ncdb:
    image: mariadb
    container_name: db
    command: --transaction-isolation=READ-COMMITTED --binlog-format=ROW
    restart: unless-stopped
    labels:
       - traefik.enable=false
```

```
      - traefik.docker.network=traefik
    volumes:
      - ncdb:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=Contrasenna
      - MYSQL_PASSWORD=Contrasenna
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
    networks:
      - nc_back

  nextcloud:
    image: nextcloud
    container_name: nextcloud
    depends_on:
      - ncdb
    volumes:
      - nextcloud:/var/www/html
    labels:
      - "traefik.enable=true"
      - "traefik.port=80"
      - "traefik.backend=nextcloud"
#      - "traefik.frontend.rule=Host:${SERV}; PathPrefixStrip:/nextcloud"
      - "traefik.frontend.rule=Host:nextcloud.${SERV}"
      - "traefik.docker.network=traefik"
    environment:
      - NEXTCLOUD_ADMIN_USER=Pablo
      - NEXTCLOUD_ADMIN_PASSWORD=Contrasenna
      - SMTP_HOST=smtp.ehu.eus
      - LC_ALL=C.UTF-8
      - TZ=Europe/Madrid
    restart: unless-stopped
    networks:
      - traefik
      - nc_back
networks:
  nc_back:
    driver: bridge
  traefik:
    external:
      name: traefik
```

Fichero 9.4: `docker-compose-nextcloud.yml`

Podemos ver la actividad o el *registro de mensajes* (también llamado `logs`) de uno de los contenedores que forman el servicio:

```
SERV='localhost' docker-compose -f docker-compose-nextcloud.yml logs
   nextcloud
```

En esta captura vemos qué se ejecuta en el contenedor de Nextcloud:

```
   SERV='localhost' docker-compose -f docker-compose-nextcloud.yml exec nextcloud
      bash
root@4134890375:/var/www/html# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.4 280804 36112 ?        Ss   17:13   0:00 apache2 -DFOREGROUND
www-data   112  0.0  0.3 281568 28032 ?        S    17:13   0:00 apache2 -DFOREGROUND
www-data   113  0.0  0.1 280836  9156 ?        S    17:13   0:00 apache2 -DFOREGROUND
www-data   114  0.0  0.1 280836  9156 ?        S    17:13   0:00 apache2 -DFOREGROUND
www-data   115  0.0  0.1 280836  9156 ?        S    17:13   0:00 apache2 -DFOREGROUND
www-data   116  0.0  0.1 280836  9156 ?        S    17:13   0:00 apache2 -DFOREGROUND
www-data   117  0.0  0.1 280836  9160 ?        S    17:14   0:00 apache2 -DFOREGROUND
root       118 13.3  0.0   5620  3540 pts/0    Ss   18:07   0:00 bash
root       124  0.0  0.0   9392  3108 pts/0    R+   18:07   0:00 ps aux
root@4134890375cd:/var/www/html#
```

A continuación, el resultado de `history` después de hacer el laboratorio:

```
 1875  ls -l
 1876  docker pull ubuntu
 1877  docker images
 1878  docker run --rm ubuntu uname -a
 1879  uname -a
 1880  docker run --rm it ubuntu bash
 1881  ps aux|grep sleep
 1882  docker run --rm -it ubuntu bash
 1883  docker ps -a
 1884  docker run ubuntu ps aux
 1885  docker ps -a
 1886  docker rm gifted_edison
 1887  docker images
 1888  docker rmi ubuntu emilevauge/whoami
 1889  docker images
 1890  ls -l
 1891  cd busybox/
 1892  l
 1893  cat Dockerfile
 1894  cp /bin/busybox .
 1895  docker build --tag ejemplo .
 1896  docker images
 1897  docker run --rm ejemplo ./busybox ls -l
 1898  docker run --rm ejemplo ./busybox ps aux
 1899  docker inspect ejemplo
 1902  sudo ls -l /var/lib/docker/overlay2/d7328064975c0...82c2857c9bdbbb2e0/diff
 1903  docker run -it --rm --name "Ejdocker" -h "ED" ejemplo ./busybox ash
 1904  docker run -dit --name web -p 8080:80 -v sitio:/usr/local/apache2/htdocs/ httpd
 1905  curl localhost:8080
 1906  docker ps -a
 1907  docker volume ls
 1908  docker volume inspect sitio
 1909  sudo ls -l /var/lib/docker/volumes/sitio/_data
 1910  sudo vi /var/lib/docker/volumes/sitio/_data/index.html
 1911  curl localhost:8080
 1912  docker stop web
 1913  docker ps -a
 1914  docker rm web
 1915  docker ps -a
 1916  docker run -dit --name web -p 8080:80 -v sitio:/usr/local/apache2/htdocs/ httpd
 1917  curl localhost:8080
```

```
1918  cd ..
1919  docker stop web
1920  docker rm web
1921  ls
1922  cd nextcloud
1923  ls
1924  cat docker-compose.yml
1925  ls
1926  docker-compose up -d
1927  curl localhost:88
1928  docker-compose ps
1929  docker ps -a
1930  docker-compose logs
1932  docker-compose logs app
1933  docker-compose down
1934  docker volume ls
1935  docker volume prune
1936  cd ..
1937  ls
1938  cd Traefik/
1939  ls
1943  docker network create traefik
1944  SERV='localhost' docker-compose -f traefik.yml up -d --scale whoami=2
1945  docker-compose -f traefik.yml ps
1946  curl localhost/whoami
1947  curl localhost/whoami1947  docker-compose -f traefik.yml ps
1948  ls
1950  SERV='localhost' docker-compose -f docker-compose-wp.yml up -d
1953  curl -H Host:portainer.localhost http://127.0.0.1
1954  curl -H Host:wp.localhost http://wp.localhost/
1955  docker-compose -f docker-compose-wp.yml logs wordpress
1956  SERV='localhost' docker-compose -f docker-compose-nextcloud.yml up -d
1958  docker-compose -f docker-compose-nextcloud.yml logs nextcloud
1959  docker-compose -f docker-compose-nextcloud.yml logs db
1965  curl -H Host:nextcloud.localhost http://nextcloud.localhost/
1966  docker ps -a
1967  SERV='localhost' docker-compose -f docker-compose-nextcloud.yml logs nextcloud
1968  SERV='localhost' docker-compose -f docker-compose-nextcloud.yml exec netxcloud bash
1973  docker-compose -f traefik.yml down
1976  docker-compose -f docker-compose-nextcloud.yml down
1977  docker-compose -f docker-compose-wp.yml down
1979  docker volume prune
1980  docker volume ls
1983  docker ps -a
1984  docker network ls
1985  docker network rm traefik
```

Fichero 9.5: `history`

9.6. Vagrant con `docker-compose`

Vagrant es un sistema de automatización de máquinas virtuales (MV), que solicita acciones a un hipervisor como *Virtualbox* para configurar una o varias MV como se define en el fichero `Vagrantfile` a partir de plantillas preinstaladas de sistemas. Por ejemplo, `jammy64` será el nombre de la `box` de Ubuntu 22.04 y si se ha descargado y está actualizada se inicia rápidamente, aunque no tanto como un contenedor.

Para el siguiente sistema no es necesario librar puertos, ya que por si acaso están usándose puertos más allá del 1024 para no necesitar permisos de `root`. Sin embargo, conviene borrar los contenedores actuales para mantener limpio el sistema, como se hace a partir de la línea 1973 del `history`.

Se puede combinar el uso de Máquinas Virtuales automatizadas con Vagrant para lanzar diferentes grupos de contenedores con `docker-compose` como se ve en el ejemplo de la carpeta *Vagrant*.

En el `Vagrantfile` se define un programa `bash` (llamados `scripts`) que instalará el software `docker-compose`, creará una red definida por software (SDN) llamada `traefik` y lanzará los tres ficheros `docker-compose`.

Después de configurar Virtualbox se define la máquina virtual con la imagen Bionic, IP por DHCP, abre puertos, *provisiona* (instala) Docker, copia los tres ficheros `docker-compose` del apartado anterior y ejecuta el `script` (en el último comentario del fichero está la alternativa sin script, usando la instrucción `inline`).

Tras ejecutar el `vagrant up`, como comprobación deberíamos acceder desde la línea de comandos con esta instruccióna Nextcloud y a Træfik:

```
curl -H Host:nextcloud.mv http://nextcloud.localhost:8000/
curl -H Host:mv http://nextcloud.localhost:8000/traefik/
```

Quedaría configurar Landrush (con `vagrant plugin install landrush`) para crear dominios dinámicos para las máquinas virtuales (necesita `root`), y habilitarlo en el `Vagrantfile` en la línea:

```
m.landrush.enabled = true # vagrant plugin install landrush
```

Éste es el `Vagrantfile`:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# This script to install docker-compose will get executed after we have
   provisioned the box
$script = <<-SCRIPT
curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-
   compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
# docker-compose up -d
docker network create traefik
SERV='mv' docker-compose -f traefik.yml up -d --scale whoami=2
SERV='mv' docker-compose -f docker-compose-wp.yml up -d # --scale wordpress
   =3 --scale db=2
SERV='mv' docker-compose -f docker-compose-nextcloud.yml up -d
SCRIPT

Vagrant.configure("2") do |config|
```

```
    config.vm.provider :virtualbox do |v|
        # On VirtualBox, we don't have guest additions or a functional
           vboxsf
        # in CoreOS, so tell Vagrant that so it can be smarter.
        v.check_guest_additions = false
        v.functional_vboxsf     = false
  v.customize ["modifyvm", :id, "--audio", "none"]
   end
end

Vagrant.configure("2") do |config|
    config.vm.define vm_name = "mv" do |m|
        m.vm.box = "ubuntu/bionic64"
 #       m.landrush.enabled = true #vagrant plugin install landrush
#        m.vm.hostname = "myhost.vagrant.test"
 #       m.cache.scope = :box   # vagrant plugin install vagrant-cachier
        m.vm.hostname = vm_name
        # ip = "172.21.12.66"
        # m.vm.network :private_network, ip: ip
      m.vm.network "private_network", type: "dhcp"
        m.vm.network "forwarded_port", guest: 80, host: 8000 # <1024
           necesita root
  #       m.vm.network "forwarded_port", guest: 8080, host: 8880
        m.vm.provision "docker"
        m.vm.provision "file", source: "./docker-compose-wp.yml",
           destination: "./docker-compose-wp.yml"
        m.vm.provision "file", source: "./docker-compose-nextcloud.yml",
           destination: "./docker-compose-nextcloud.yml"
        m.vm.provision "file", source: "./traefik.yml", destination: "./
           traefik.yml"
        m.vm.provision "shell", inline: $script
#        m.vm.provision "shell",
#inline: "DEBIAN_FRONTEND=noninteractive SERV='mv' docker-compose -f
   traefik.yml up -d --scale whoami=2 && SERV='mv' docker-compose -f docker
   -compose-wp.yml -d && SERV='mv' docker-compose -f docker-compose-
   nextcloud.yml -d
    end
end
```

Fichero 9.6: `Vagrantfile`