

# Despliegue de un Multicontenedor en Azure

Informática  
como Servicio

SOFÍA ALEJANDRA LÓPEZ FERNÁNDEZ

Grupo 1.1  
Máster Universitario en Ingeniería Informática

# Índice

<b>1. Información de la Práctica</b>	<b>3</b>
<b>2. Pasos previos al despliegue</b>	<b>3</b>
2.1. /php . . . . .	3
2.2. /apache . . . . .	4
2.3. php/src y apache/src . . . . .	4
2.4. Fichero docker-compose.yml . . . . .	5
<b>3. Despliegue de la Aplicación en local</b>	<b>6</b>
3.1. Instalación Docker y Compose . . . . .	6
3.2. Ejecución de la aplicación . . . . .	7
<b>4. Despliegue de la Aplicación en Azure</b>	<b>9</b>
4.1. Elementos para el despliegue . . . . .	9
4.2. Ejecución del despliegue . . . . .	9
4.2.1. Notas: . . . . .	16

# Índice de figuras

1. Ejecución del Compose - Inicio. . . . .	8
2. Ejecución del Compose - Final. . . . .	8
3. Verificación de creación de imágenes y puesta en marcha de contenedores. . .	9
4. Creación del Grupo de Recursos. . . . .	10
5. Creación del Registro de Contenedor. . . . .	11
6. Creación del Plan de App Service. . . . .	12
7. Tags de las imágenes. . . . .	12
8. Push de las imágenes al Registro de Contenedor. . . . .	13
9. Verificación de subida de los contenedores. . . . .	13
10. Creación de la Aplicación web. . . . .	14
11. Configuración de Logs de la aplicación. . . . .	15
12. Configuración de la aplicación. . . . .	16
13. Creación de un usuario. . . . .	17
14. Creación de un usuario existente. . . . .	17
15. Inicio de sesión de un usuario. . . . .	18

## 1. Información de la Práctica

El despliegue que se va a llevar a cabo en Azure es el de **tres** imágenes docker correspondientes a Apache, PHP y MySQL.

Para poder realizar este despliegue se ha creado un archivo `docker-compose.yml` y así poder lanzar los tres contenedores a la vez, junto con sus dependencias.

Se han escogido las siguientes versiones de los contenedores:

- **Apache:** versión 2.4.33 del servidor Apache para la imagen linux alpine, para obtener una imagen lo menos pesada posible. El proyecto Alpine Linux proporciona una imagen de menor tamaño que las imágenes generales.
- **PHP:** versión 7.2.7 FPM. PHP-FPM proporciona una implementación FastCGI.
- **MySQL:** versión 5.6.40, teniendo en cuenta que otras versiones de MySQL han dado problemas a la hora de conectar con otros contenedores, se ha preferido utilizar esa implementación más estable.

## 2. Pasos previos al despliegue

Antes de realizar el despliegue de nuestra web, primero se tienen que crear los archivos de configuración de Docker y el archivo `docker-compose`. Para ello, se debe crear el siguiente árbol de directorios:

```
miweb
├── php
│   └── src
└── apache
    └── src
```

Cada una de esas carpetas tendrá la información que se explicará a continuación:

### 2.1. /php

Contendrá un archivo `Dockerfile` con el que se generará la imagen PHP correspondiente:

```
# Este es el archivo php/Dockerfile
FROM php:7.2.7-fpm-alpine3.7
# Obtiene la informacin del directorio src que se encuentra en este directorio
COPY src/ /var/www/html/
# Actualiza los paquetes
RUN apk update; \
    apk upgrade;
# Instala mysqli para facilitar las conexiones con MySQL
RUN docker-php-ext-install mysqli
```

## 2.2. /apache

Contendrá un archivo ‘Dockerfile’ con el que se generará la imagen Apache correspondiente y un archivo de configuración de Apache ‘apache.conf’ para manejar las peticiones de archivos PHP a su propio contenedor, también se establecen las configuraciones de los módulos proxy necesarios para el correcto funcionamiento de este.

```
# Este es el archivo apache/Dockerfile
FROM httpd:2.4.33-alpine
# Actualiza los paquetes
RUN apk update; \
    apk upgrade;
# Copia el archivo de configuracin apache
COPY apache.conf /usr/local/apache2/conf/apache.conf
# Obtiene la informacin del directorio src que se encuentra en la raiz miweb/
COPY src/ /var/www/html/
# Corre el archivo de configuracion previamente copiado
RUN echo "Include /usr/local/apache2/conf/apache.conf" \
    >> /usr/local/apache2/conf/httpd.conf
```

```
#Este es el archivo apache/apache.conf
ServerName localhost

LoadModule deflate_module /usr/local/apache2/modules/mod_deflate.so
LoadModule proxy_module /usr/local/apache2/modules/mod_proxy.so
LoadModule proxy_fcgi_module /usr/local/apache2/modules/mod_proxy_fcgi.so

<VirtualHost *:80>
# Peticiones Proxy .php al puerto 9000 del contenedor php-fpm
ProxyPassMatch ^/(.*\.php(/.*?))$ fcgi://php:9000/var/www/html/$1
DocumentRoot /var/www/html/
<Directory /var/www/html/>
    DirectoryIndex index.php
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>

# Envio de logs
CustomLog /proc/self/fd/1 common
ErrorLog /proc/self/fd/2
</VirtualHost>
```

## 2.3. php/src y apache/src

Este directorio contendrá toda la información de la página web que se va a desplegar. En el caso de esta práctica, tiene los siguientes archivos:

```

miweb/apache/src/ ó /miweb/php/src/
├─ index.php  Pagina de inicio de la aplicación
├─ inicio.php  Pagina de sesion ya iniciada
├─ indexjs.js  JS con el funcionamiento de las páginas
├─ css  Fichero de estilos.
├─ js  Ficheros de comportamiento Bootstrap y JQuery
├─ php  Ficheros PHP que obtienen la información del servidor

```

## 2.4. Fichero docker-compose.yml

A mayores de los anteriores directorios, se tiene que crear el fichero `docker-compose.yml` para poder ejecutar los tres contenedores al mismo tiempo, junto con sus dependencias. Aunque teniendo en cuenta que esta aplicación se va a desplegar en Azure, se deberá crear un segundo fichero con las modificaciones necesarias para su despliegue en Azure. El siguiente script refleja las configuraciones necesarias para el fichero `docker-compose.yml`:

```

# La version 3 es la recomendada en Compose File
version: "3.2"
# Se declaran todos los servicios que se van a montar a partir de este archivo
services:
  php:
    build: './php/'
    networks:
      - mired
  apache:
    build: './apache/'
    depends_on:
      - php
      - mysql
    networks:
      - mired
    ports:
      - "8080:80"
  mysql:
    image: mysql:5.6.40
    networks:
      - mired
    environment:
      - MYSQL_ROOT_PASSWORD=rootpassword
# Se declaran las redes en las que se van a desplegar los contenedores
networks:
  mired:

```

En el caso del archivo `docker-compose-azure.yml`, se modificarán aquellas variables necesarias para su correcto funcionamiento:

```
version: "3.2"
services:
  php:
    image: <nombre-container-registry>.azurecr.io/php:<tag>
    networks:
      - mired
  apache:
    image: <nombre-container-registry>.azurecr.io/apache:<tag>
    depends_on:
      - php
      - mysql
    networks:
      - mired
    ports:
      - "8080:80"
  mysql:
    image: <nombre-container-registry>.azurecr.io/mysql:<tag>
    networks:
      - mired
    environment:
      - MYSQL_ROOT_PASSWORD=rootpassword
networks:
  mired:
\end
```

Donde cada una de las imágenes se encontrará asociada al Registro de Contenedor creado donde se ha realizado el push, con el siguiente formato `<nombre-container-registry>.azurecr.io/<imagen-contenedor>:<tag>`. En este caso, se va a crear un *Registro de Contenedor* llamado **miPlanServicioSofiaL** y todas las imágenes se corresponderán a la versión 1.

## 3. Despliegue de la Aplicación en local

### 3.1. Instalación Docker y Compose

Para realizar el despliegue en local se debe tener instalado Docker, véase **ICS. PaaS. Configuración del entorno.pdf**, y Docker compose. Para ello, se debe insertar el siguiente comando (última versión <https://github.com/docker/compose/releases>):

```
curl -L https://github.com/docker/compose/releases/download/1.25.1-rc1/docker-compose-`
  uname -s`-`uname -m` -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

Listing 1: bash version

Para comprobar que se ha instalado correctamente y la versión con la que se está trabajando, se debe introducir el siguiente comando:

```
docker-compose --version
```

Listing 2: bash version

En el caso de mi máquina, obtengo el siguiente resultado:

```
sofialf@sofialf-VirtualBox:~$ docker-compose --version
docker-compose version 1.25.1-rc1, build d92e9bee
```

Listing 3: bash version

## 3.2. Ejecución de la aplicación

Para obtener todo lo necesario para ejecutar en local la aplicación, se debe clonar el repositorio público creado para esta práctica en el directorio deseado:

```
git clone https://github.com/Salf1997/practica-docker-muei-sofialopez
cd practica-docker-muei-sofialopez
```

Listing 4: bash version

Dentro de este directorio se encuentran los siguientes archivos y subdirectorios:

```
sofialf@sofialf-VirtualBox:~/ics-docker/practica-docker-muei-sofialopez$ ls -la
total 24
drwxr-xr-x 4 root root 4096 dic 4 01:37 .
drwxr-xr-x 11 root root 4096 dic 4 01:37 ..
drwxr-xr-x 8 root root 4096 dic 4 01:37 .git
-rw-r--r-- 1 root root 66 dic 4 01:37 .gitattributes
drwxr-xr-x 5 root root 4096 dic 4 01:37 miweb
-rw-r--r-- 1 root root 147 dic 4 01:37 readme.md
```

Ahora se debe acceder al directorio **\*\*miweb\*\*** y ejecutar el siguiente comando para lanzar los contenedores:

```
sudo docker-compose up -d
```

Listing 5: bash version

El parámetro *-d* deja en segundo plano ejecutando la aplicación, equivalente al parámetro *-detach*. La ejecución de ese comando daría el siguiente resultado:

```

Archivo Editar Ver Buscar Terminal Ayuda
root@sofialf-VirtualBox:/home/sofialf/ics-docker/practica-docker-muel-sofialopez/miweb# docker-compose up -d
Building php
Step 1/4 : FROM php:7.2.7-fpm-alpine3.7
----> 9cf17fea14c0
Step 2/4 : COPY src/ /var/www/html/
----> fd516143ee37
Step 3/4 : RUN apk update; apk upgrade;
----> Running in 6ec0ec956c94
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/community/x86_64/APKINDEX.tar.gz
v3.7.3-176-g91cc5184e1 [http://dl-cdn.alpinelinux.org/alpine/v3.7/main]
v3.7.3-165-g0f7ecd696d [http://dl-cdn.alpinelinux.org/alpine/v3.7/community]
OK: 9066 distinct packages available
Upgrading critical system libraries and apk-tools:
(1/1) Upgrading apk-tools (2.9.1-r2 -> 2.10.1-r0)
Executing busybox-1.27.2-r11.trigger
Continuing the upgrade transaction with new apk-tools:
(1/11) Upgrading musl (1.1.18-r3 -> 1.1.18-r4)
(2/11) Upgrading ca-certificates (20171114-r0 -> 20190108-r0)
(3/11) Upgrading libssh2 (1.8.0-r2 -> 1.9.0-r1)
(4/11) Upgrading libcurl (7.60.0-r1 -> 7.61.1-r3)
(5/11) Upgrading curl (7.60.0-r1 -> 7.61.1-r3)
(6/11) Upgrading tar (1.29-r1 -> 1.32-r0)
(7/11) Upgrading ncurses-terminfo-base (6.0_p20171125-r0 -> 6.0_p20171125-r1)
(8/11) Upgrading ncurses-terminfo (6.0_p20171125-r0 -> 6.0_p20171125-r1)
(9/11) Upgrading ncurses-libs (6.0_p20171125-r0 -> 6.0_p20171125-r1)
(10/11) Upgrading libxml2 (2.9.7-r0 -> 2.9.8-r1)
(11/11) Upgrading musl-utils (1.1.18-r3 -> 1.1.18-r4)
Executing busybox-1.27.2-r11.trigger
Executing ca-certificates-20190108-r0.trigger
OK: 16 MiB in 29 packages
Removing intermediate container 6ec0ec956c94
----> af69899e531c
Step 4/4 : RUN docker-php-ext-install mysqli
----> Running in 3c2b0e2f2b0c
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/community/x86_64/APKINDEX.tar.gz

```

Figura 1: Ejecución del Compose - Inicio.

```

Archivo Editar Ver Buscar Terminal Ayuda
(8/24) Upgrading ncurses-terminfo-base (6.0_p20171125-r0 -> 6.0_p20171125-r1)
(9/24) Upgrading ncurses-terminfo (6.0_p20171125-r0 -> 6.0_p20171125-r1)
(10/24) Upgrading ncurses-libs (6.0_p20171125-r0 -> 6.0_p20171125-r1)
(11/24) Upgrading expat (2.2.5-r0 -> 2.2.8-r0)
(12/24) Upgrading expat-dev (2.2.5-r0 -> 2.2.8-r0)
(13/24) Upgrading libldap (2.4.45-r3 -> 2.4.48-r0)
(14/24) Upgrading openldap-dev (2.4.45-r3 -> 2.4.48-r0)
(15/24) Upgrading sqlite-libs (3.21.0-r1 -> 3.25.3-r2)
(16/24) Upgrading sqlite-dev (3.21.0-r1 -> 3.25.3-r2)
(17/24) Upgrading libpq (10.4-r0 -> 10.10-r0)
(18/24) Upgrading postgresql-libs (10.4-r0 -> 10.10-r0)
(19/24) Upgrading postgresql-dev (10.4-r0 -> 10.10-r0)
(20/24) Upgrading libbz2 (1.0.6-r6 -> 1.0.6-r7)
(21/24) Upgrading perl (5.26.2-r1 -> 5.26.3-r0)
(22/24) Upgrading nghttp2-libs (1.28.0-r0 -> 1.39.2-r0)
(23/24) Upgrading libxml2 (2.9.7-r0 -> 2.9.8-r1)
(24/24) Upgrading musl-utils (1.1.18-r3 -> 1.1.18-r4)
Executing busybox-1.27.2-r11.trigger
OK: 91 MiB in 54 packages
Removing intermediate container 680a826ecbaa
----> fd69dc943511
Step 3/5 : COPY apache.conf /usr/local/apache2/conf/apache.conf
----> 78fae915dd59
Step 4/5 : COPY src/ /var/www/html/
----> 01b678967ce0
Step 5/5 : RUN echo "Include /usr/local/apache2/conf/apache.conf" >> /usr/local/apache2/conf/httpd.conf
----> Running in f0c2c75f8ae7
Removing intermediate container f0c2c75f8ae7
----> 7dd61d39af35
Successfully built 7dd61d39af35
Successfully tagged miweb_apache:latest
WARNING: Image for service apache was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compose up --build`.
Creating miweb_mysql_1 ... done
Creating miweb_php_1 ... done
Creating miweb_apache_1 ... done
root@sofialf-VirtualBox:/home/sofialf/ics-docker/practica-docker-muel-sofialopez/miweb#

```

Figura 2: Ejecución del Compose - Final.

En el momento en el que se ejecuta el comando anterior, se crean las imágenes y se levantan los contenedores, como se muestra en la siguiente imagen:



```

root@sofialf-VirtualBox: /home/sofialf/ics-docker/practica-docker-muel-sofialopez/miweb# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
miweb_apache         latest              7dd61d39af35       4 minutes ago      151MB
miweb_php            latest              ee8153ec462b       5 minutes ago      90.7MB
mysql                5.6.40             50328380b2b4       16 months ago      256MB
php                  7.2.7-fpm-alpine3.7 9cf17fea14c0       17 months ago      78.3MB
httpd                 2.4.33-alpine       73a557ff177a       17 months ago      91.3MB
cloudera/quickstart  latest              4239cd2958c6       3 years ago         6.34GB
root@sofialf-VirtualBox: /home/sofialf/ics-docker/practica-docker-muel-sofialopez/miweb# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
770ae7481682       miweb_apache       "httpd-foreground"  5 minutes ago      Up 5 minutes        0.0.0.0:8080->80/tcp  miweb_apache_1
60a2bde6aa50       miweb_php          "docker-php-entrypoi..."  5 minutes ago      Up 5 minutes        9000/tcp            miweb_php_1
fb034cc0633b       mysql:5.6.40       "docker-entrypoint.s..."  5 minutes ago      Up 5 minutes        3306/tcp            miweb_mysql_1
root@sofialf-VirtualBox: /home/sofialf/ics-docker/practica-docker-muel-sofialopez/miweb#

```

Figura 3: Verificación de creación de imágenes y puesta en marcha de contenedores.

Para ver que se ha lanzado con éxito la aplicación en local, se debe acceder a `http://localhost:8080/`.

## 4. Despliegue de la Aplicación en Azure

### 4.1. Elementos para el despliegue

Para realizar en despliegue en Azure se van a utilizar el *Grupo de Recursos* y el *Registro de Contenedor* creados en la práctica ICS. **PaaS. Despliegue imagen docker sobre Azure desde consola Ubuntu.pdf**. Además de estos elementos que proporciona Azure, se utilizarán:

- **Plan de App Service:** permite crear sitios web para cualquier plataforma, se utilizará el plan Basic que tiene un coste 0,016€/hora (B1) y se creará en un contenedor Linux.
- **Aplicación web:** como se va a crear una aplicación multi-contenedor, esta es la opción que nos proporcionar Azure.

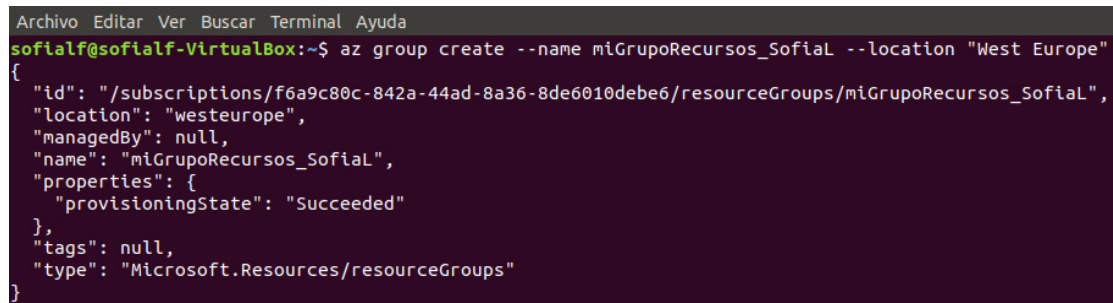
### 4.2. Ejecución del despliegue

Para llevar a cabo la configuración del despliegue se van a seguir los siguientes pasos:

1. Crear el Grupo de Recursos:

```
az group create --name miGrupoRecursos_SofiaL --location "West Europe"
```

Listing 6: bash version



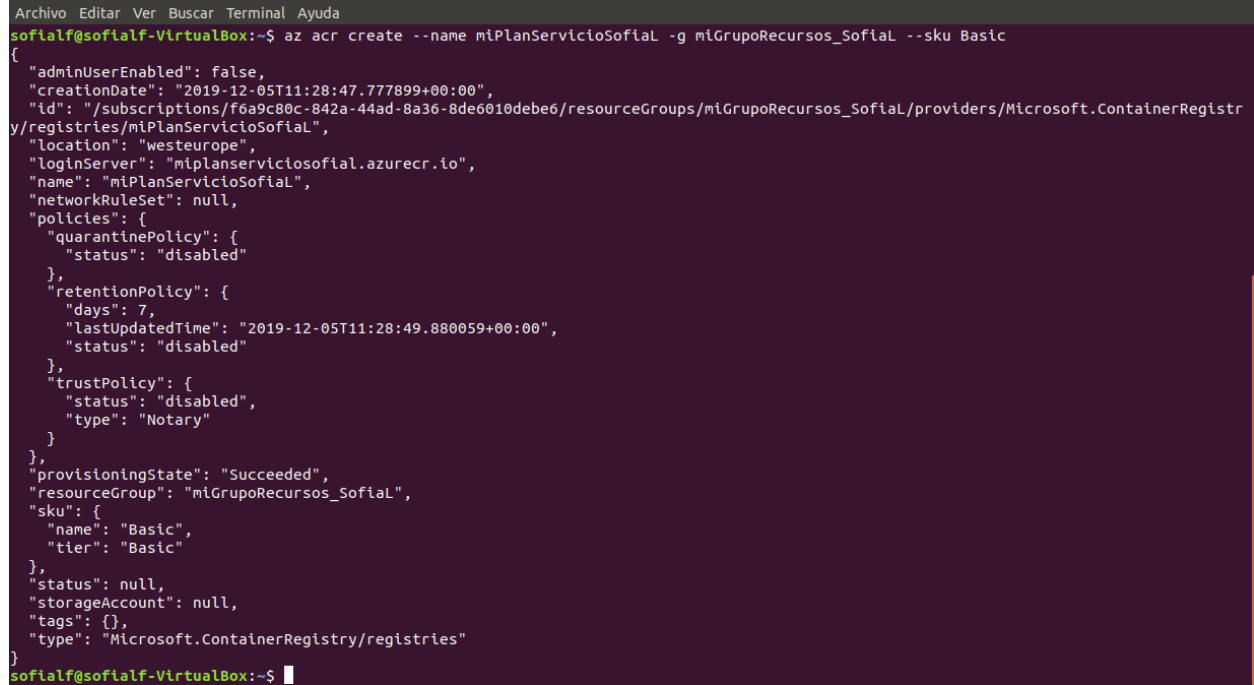
```
Archivo Editar Ver Buscar Terminal Ayuda
sofia1f@sofia1f-VirtualBox:~$ az group create --name miGrupoRecursos_SofiaL --location "West Europe"
{
  "id": "/subscriptions/f6a9c80c-842a-44ad-8a36-8de6010debe6/resourceGroups/miGrupoRecursos_SofiaL",
  "location": "westeurope",
  "managedBy": null,
  "name": "miGrupoRecursos_SofiaL",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

Figura 4: Creación del Grupo de Recursos.

## 2. Crear el Registro de Contenedor:

```
az acr create --name miPlanServicioSofiaL -g miGrupoRecursos_SofiaL --sku Basic
```

Listing 7: bash version



```

Archivo Editar Ver Buscar Terminal Ayuda
sofialf@sofialf-VirtualBox:~$ az acr create --name miPlanServicioSofiaL -g miGrupoRecursos_SofiaL --sku Basic
{
  "adminUserEnabled": false,
  "creationDate": "2019-12-05T11:28:47.777899+00:00",
  "id": "/subscriptions/f6a9c80c-842a-44ad-8a36-8de6010debe6/resourceGroups/miGrupoRecursos_SofiaL/providers/Microsoft.ContainerRegistr
y/registries/miPlanServicioSofiaL",
  "location": "westeurope",
  "loginServer": "miplanserviciosofial.azurecr.io",
  "name": "miPlanServicioSofiaL",
  "networkRuleSet": null,
  "policies": {
    "quarantinePolicy": {
      "status": "disabled"
    },
    "retentionPolicy": {
      "days": 7,
      "lastUpdatedTime": "2019-12-05T11:28:49.880059+00:00",
      "status": "disabled"
    },
    "trustPolicy": {
      "status": "disabled",
      "type": "Notary"
    }
  },
  "provisioningState": "Succeeded",
  "resourceGroup": "miGrupoRecursos_SofiaL",
  "sku": {
    "name": "Basic",
    "tier": "Basic"
  },
  "status": null,
  "storageAccount": null,
  "tags": {},
  "type": "Microsoft.ContainerRegistry/registries"
}
sofialf@sofialf-VirtualBox:~$

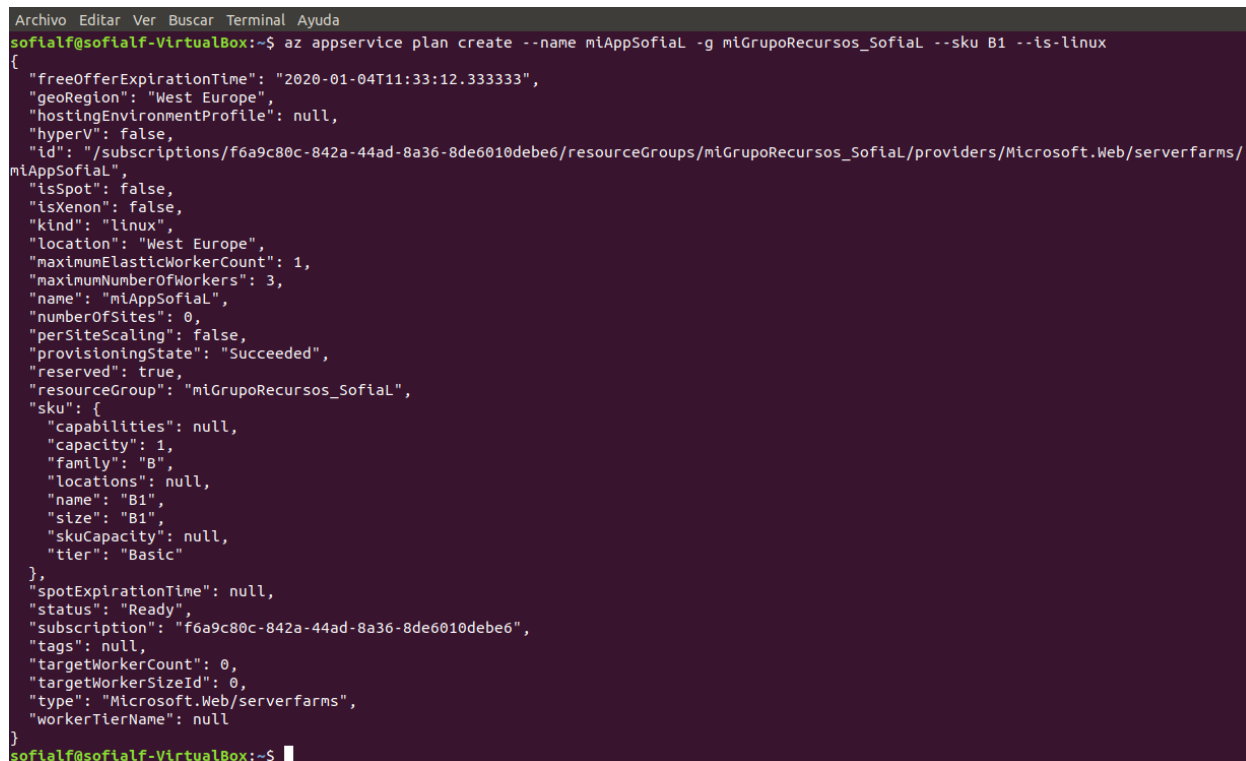
```

Figura 5: Creación del Registro de Contenedor.

### 3. Crear Plan de App Service:

```
az appservice plan create --name miAppSofiaL -g miGrupoRecursos_SofiaL --sku B1 --is-linux
```

Listing 8: bash version

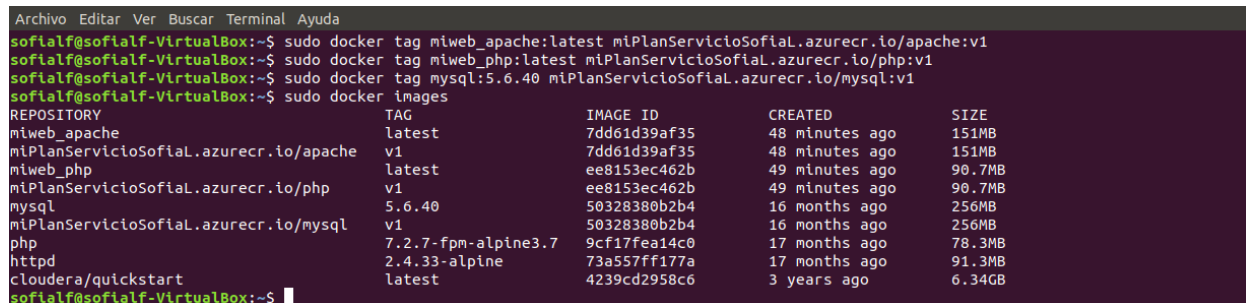


```
Archivo Editar Ver Buscar Terminal Ayuda
sofialf@sofialf-VirtualBox:~$ az appservice plan create --name miAppSofiaL -g miGrupoRecursos_SofiaL --sku B1 --is-linux
{
  "freeOfferExpirationTime": "2020-01-04T11:33:12.333333",
  "geoRegion": "West Europe",
  "hostingEnvironmentProfile": null,
  "hyperV": false,
  "id": "/subscriptions/f6a9c80c-842a-44ad-8a36-8de6010debe6/resourceGroups/miGrupoRecursos_SofiaL/providers/Microsoft.Web/serverfarms/miAppSofiaL",
  "isSpot": false,
  "isXenon": false,
  "kind": "linux",
  "location": "West Europe",
  "maximumElasticWorkerCount": 1,
  "maximumNumberOfWorkers": 3,
  "name": "miAppSofiaL",
  "numberOfSites": 0,
  "perSiteScaling": false,
  "provisioningState": "Succeeded",
  "reserved": true,
  "resourceGroup": "miGrupoRecursos_SofiaL",
  "sku": {
    "capabilities": null,
    "capacity": 1,
    "family": "B",
    "locations": null,
    "name": "B1",
    "size": "B1",
    "skuCapacity": null,
    "tier": "Basic"
  },
  "spotExpirationTime": null,
  "status": "Ready",
  "subscription": "f6a9c80c-842a-44ad-8a36-8de6010debe6",
  "tags": null,
  "targetWorkerCount": 0,
  "targetWorkerSizeId": 0,
  "type": "Microsoft.Web/serverfarms",
  "workerTierName": null
}
sofialf@sofialf-VirtualBox:~$
```

Figura 6: Creación del Plan de App Service.

### 4. Push de las imágenes al Registro de Contenedor:

Se hace un tag por cada una de las imágenes que se van a utilizar en la web y se hace el push de esa imagen local al despliegue en Azure.



```
Archivo Editar Ver Buscar Terminal Ayuda
sofialf@sofialf-VirtualBox:~$ sudo docker tag miweb_apache:latest miPlanServicioSofiaL.azurecr.io/apache:v1
sofialf@sofialf-VirtualBox:~$ sudo docker tag miweb_php:latest miPlanServicioSofiaL.azurecr.io/php:v1
sofialf@sofialf-VirtualBox:~$ sudo docker tag mysql:5.6.40 miPlanServicioSofiaL.azurecr.io/mysql:v1
sofialf@sofialf-VirtualBox:~$ sudo docker images
REPOSITORY                                TAG                IMAGE ID           CREATED           SIZE
miweb_apache                             latest            7dd61d39af35      48 minutes ago   151MB
miPlanServicioSofiaL.azurecr.io/apache    v1               7dd61d39af35      48 minutes ago   151MB
miweb_php                                 latest            ee8153ec462b      49 minutes ago   90.7MB
miPlanServicioSofiaL.azurecr.io/php        v1               ee8153ec462b      49 minutes ago   90.7MB
mysql                                       5.6.40           50328380b2b4      16 months ago    256MB
miPlanServicioSofiaL.azurecr.io/mysql      v1               50328380b2b4      16 months ago    256MB
php                                         7.2.7-fpm-alpine3.7 9cf17fea14c0      17 months ago    78.3MB
httpd                                       2.4.33-alpine     73a557ff177a      17 months ago    91.3MB
cloudera/quickstart                       latest            4239cd2958c6      3 years ago       6.34GB
sofialf@sofialf-VirtualBox:~$
```

Figura 7: Tags de las imágenes.

```

Archivo Editar Ver Buscar Terminal Ayuda
sofialf@sofialf-VirtualBox:~$ sudo docker push miPlanServicioSofial.azurecr.io/apache:v1
The push refers to repository [miPlanServicioSofial.azurecr.io/apache]
e66e6fe4b337: Pushed
2ac5ced55ee3: Pushed
4b458661f264: Pushed
cbae572d11bf: Pushed
27c06e86c5e0: Pushed
382d19e39fcd: Pushed
d396b598d25a: Pushed
10782475242d: Pushed
717b092b8c86: Pushed
v1: digest: sha256:84efec817d43f4eb10878b5682754b574615a3c5cedf7be7717cd4e62ad18cc0 size: 2199
sofialf@sofialf-VirtualBox:~$ sudo docker push miPlanServicioSofial.azurecr.io/php:v1
The push refers to repository [miPlanServicioSofial.azurecr.io/php]
b7d6cc909a2a: Pushed
35d3f39ab16d: Pushed
e273e8538ad3: Pushed
832ac63d3765: Pushed
3832dda763c7: Pushed
f51c0aeaa90: Pushed
2913a16d78a9: Pushed
313f5a50baab: Pushed
a80759dc8ee5: Pushed
26e7e38f6501: Pushed
716ded580180: Pushed
b121fd9a15c8: Pushed
1d57d892ec64: Pushed
717b092b8c86: Mounted from apache
v1: digest: sha256:8a2f9e9a0c15698053143c26d19688073c7d790d3283cf7bb15bbb4ae1eecdab size: 3249
sofialf@sofialf-VirtualBox:~$ sudo docker push miPlanServicioSofial.azurecr.io/mysql:v1
The push refers to repository [miPlanServicioSofial.azurecr.io/mysql]
a039fbca3810: Pushed
97543b3cd3d3: Pushed
6135c8925f31: Pushed
7327252b6f03: Pushed
912b35b094bc: Pushed
a9d3f9d0dda: Pushed
c0c26734fb83: Pushed
4801a487d51a: Pushed
aae63f31dee9: Pushed
6f8d38b0e2b6: Pushed
cdb3f9544e4c: Pushed
v1: digest: sha256:63638985e606efa6f5059948c8c1185651882e1309457c069f6ac244a386672a size: 2621
sofialf@sofialf-VirtualBox:~$

```

Figura 8: Push de las imágenes al Registro de Contenedor.

Se puede comprobar que las imágenes están subidas en el portal de Azure, en inicio > Grupos de Recursos > miGrupoRecursos\_SofiaL > miPlanServicioSofiaL - Repositorios.

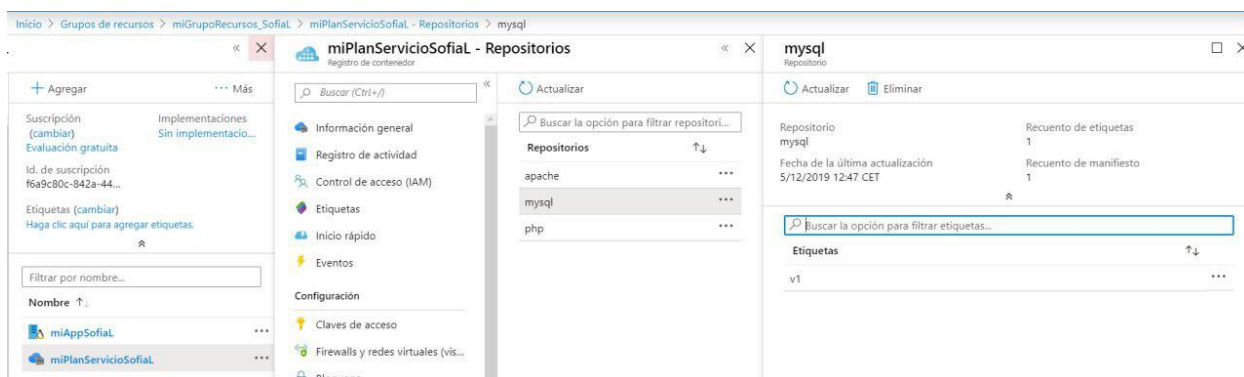


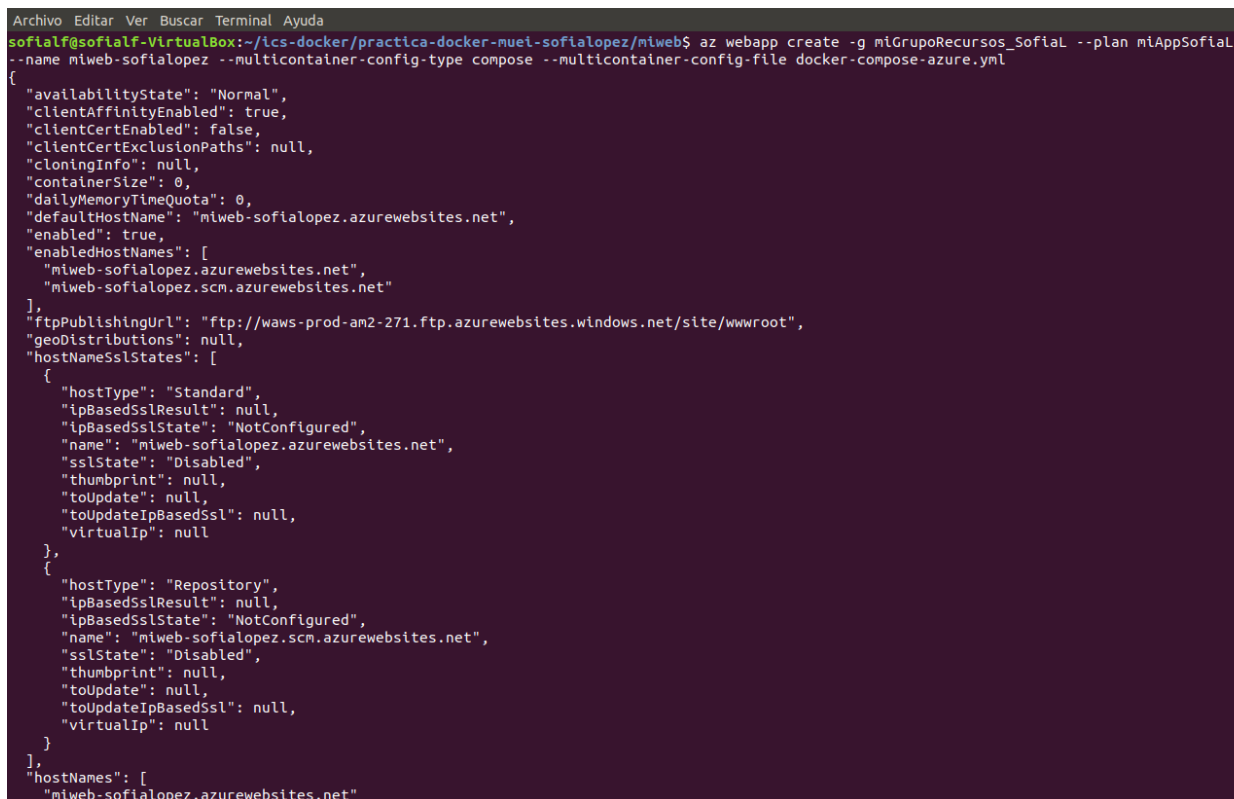
Figura 9: Verificación de subida de los contenedores.

## 5. Crear la Aplicación web:

Como se describió anteriormente, este despliegue se va a realizar con un Compose de docker. Para ello, se va a utilizar el archivo `docker-compose-azure.yml` que contiene la información de los contenedores adaptada para su despliegue en Azure.

```
az webapp create -g miGrupoRecursos_SofiaL --plan miAppSofiaL --name miweb-sofialopez --multicontainer-config-type compose --multicontainer-config-file docker-compose-azure.yml
```

Listing 9: bash version



```

Archivo Editar Ver Buscar Terminal Ayuda
sofia1f@sofia1f-VirtualBox:~/ics-docker/practica-docker-nuei-sofialopez/miweb$ az webapp create -g miGrupoRecursos_SofiaL --plan miAppSofiaL
--name miweb-sofialopez --multicontainer-config-type compose --multicontainer-config-file docker-compose-azure.yml
{
  "availabilityState": "Normal",
  "clientAffinityEnabled": true,
  "clientCertEnabled": false,
  "clientCertExclusionPaths": null,
  "cloningInfo": null,
  "containerSize": 0,
  "dailyMemoryTimeQuota": 0,
  "defaultHostName": "miweb-sofialopez.azurewebsites.net",
  "enabled": true,
  "enabledHostNames": [
    "miweb-sofialopez.azurewebsites.net",
    "miweb-sofialopez.scm.azurewebsites.net"
  ],
  "ftpPublishingUrl": "ftp://waws-prod-am2-271.ftp.azurewebsites.windows.net/site/wwwroot",
  "geoDistributions": null,
  "hostNameSslStates": [
    {
      "hostType": "Standard",
      "ipBasedSslResult": null,
      "ipBasedSslState": "NotConfigured",
      "name": "miweb-sofialopez.azurewebsites.net",
      "sslState": "Disabled",
      "thumbprint": null,
      "toUpdate": null,
      "toUpdateIpBasedSsl": null,
      "virtualip": null
    },
    {
      "hostType": "Repository",
      "ipBasedSslResult": null,
      "ipBasedSslState": "NotConfigured",
      "name": "miweb-sofialopez.scm.azurewebsites.net",
      "sslState": "Disabled",
      "thumbprint": null,
      "toUpdate": null,
      "toUpdateIpBasedSsl": null,
      "virtualip": null
    }
  ],
  "hostNames": [
    "miweb-sofialopez.azurewebsites.net"
  ]
}

```

Figura 10: Creación de la Aplicación web.

Se van a habilitar los logs dentro de la aplicación para poder manejar los errores que vayan apareciendo. Para ello, hay que actualizar la configuración de registros en inicio > Grupos de Recursos > miGrupoRecursos\_SofiaL > miweb-sofialopez - Registros de App Service, habilitando el *Sistema de archivos*.

 Guardar
  Descartar

---

**Registro de la aplicación** ⓘ

Desactivado
 **Sistema de archivos**

**Cuota (MB) \*** ⓘ

**Período de retención (días) \*** ⓘ

**Registros de descarga**

FTP/Nombre de usuario de implementación	No se definió ningún FTP/usuario de implementación
FTP	ftp://waws-prod-am2-271.ft.azurewebsites.windows.net
FTPS	ftps://waws-prod-am2-271.ft.azurewebsites.windows.net

Figura 11: Configuración de Logs de la aplicación.

El error que nos proporcionan los logs es el siguiente:

```
ERROR - Image pull failed: Verify docker image configuration and credentials (if
      using private repository)
ERROR - multi-container unit was not started successfully
```

Este error significa que no existe la imagen del contenedor a la que se está llamando, esto es porque se ha creado la web, pero no se ha vinculado el registro de contenedor que se creó en el **punto 2**.

#### 6. Introducir parámetros de configuración:

Para poder desplegar correctamente la aplicación, se han de introducir los siguientes parámetros:

```
az webapp config container set --docker-registry-server-password XCq=
MOXl7md6g75zKeA0yRGCWt48G5wg --docker-registry-server-url https://
miPlanServicioSofiaL.azurecr.io --docker-registry-server-user
miPlanServicioSofiaL --name miweb-sofialopez --resource-group
miGrupoRecursos_SofiaL
```

Listing 10: bash version

```
[Archivo Editar Ver Buscar Terminal Ayuda]
sofialf@sofialf-VirtualBox:~/ics-docker/practica-docker-muei-sofialopez/miweb$ az webapp config container set --docker-registry-server-passwo
rd VREcJUEymkKjxK5+S79m1HK+8RHWYWF --docker-registry-server-url https://miPlanServicioSofiaL.azurecr.io --docker-registry-server-user miPlan
ServicioSofiaL --name miweb-sofialopez --resource-group miGrupoRecursos_SofiaL
[
{
  "name": "DOCKER_REGISTRY_SERVER_URL",
  "slotSetting": false,
  "value": "https://miPlanServicioSofiaL.azurecr.io"
},
{
  "name": "DOCKER_REGISTRY_SERVER_USERNAME",
  "slotSetting": false,
  "value": "miPlanServicioSofiaL"
},
{
  "name": "DOCKER_REGISTRY_SERVER_PASSWORD",
  "slotSetting": false,
  "value": null
},
{
  "name": "DOCKER_CUSTOM_IMAGE_NAME",
  "value": "COMPOSE|dnVyc2lvbjogIjMuMiIKc2VydmnlZXM6CiAgcGhwOgogICAgaw1hZ2U6IGlpcGhbbnNlcnpZy2Lvc2xcwmcFjdGljYS5henVzZWNYLmlvL3BocDp2MQogICA
gbnVodD29ya3M6CiAgICAgIC0gbWlyZXQKICBhcGFjaGU6CiAgICBpbWFnZTogbwFlwbGuc2VydmnljaW9zbHBByYWNoaWNhNmF6dXJlY3luaW8vbXZlcw6djkEKCAGISGLHdvcmt2ogogICAgICATIGlpcnVkClAgICBlbnZpcm9ubWVuDoKICAGLSBNVNRFt9St0
9UX1BBU1NXt1JEpxJvb3RwYXNZd29yZApuzXR3b3JrczoKICBtaXJLZDoK"
}
]
sofialf@sofialf-VirtualBox:~/ics-docker/practica-docker-muei-sofialopez/miweb$
```

Figura 12: Configuración de la aplicación.

Donde `--docker-registry-server-url` requiere la dirección del *Registro de Contenedor*, `--docker-registry-server-password` pide su contraseña y `--docker-registry-server-user` requiere el nombre del registro.

A continuación, se debe reiniciar la aplicación para que se apliquen los cambios:

```
az webapp restart -n miweb-sofialopez -g miGrupoRecursos_SofiaL
```

Listing 11: bash version

Pasado un tiempo, se debe poder visualizar la aplicación correctamente. Para acceder a ella, se debe usar el siguiente enlace:

<https://miweb-sofialopez.azurewebsites.net/>

### 4.2.1. Notas:

- Durante el despliegue de la aplicación puede que la web emita un error 502 - Web server received an invalid response while acting as a gateway or proxy server, esto significa que se ha desplegado el contenedor de Apache, pero se están configurando el resto de contenedores.
- Una vez desplegada la aplicación, puede añadir un usuario con una contraseña y le saldrá el siguiente mensaje:



The screenshot shows a web browser window with the URL `https://miweb-sofialopez.azurewebsites.net`. The page has a header with the word "HOME" in green. Below the header, there are two main sections: "Inicio de Sesión" (Login) and "Registrar usuario" (Register user). The "Registrar usuario" section shows a successful registration message: "Registro realizado con éxito" (Registration completed successfully) in a green box. The "Login" section has empty input fields for "Usuario" (Username) and "Contraseña" (Password), and a green "Login" button. The "Register" section has input fields for "Usuario" (containing "admin") and "Contraseña" (containing "\*\*\*\*\*"), and a green "Registrar" button. The footer contains the text: "Sofía Alejandra López Fernández", "Informática como Servicio", and "Máster Universitario en Ingeniería Informática".

Figura 13: Creación de un usuario.

- En el caso de que quiera añadir un usuario existente, lanzaría un error:

The screenshot shows the same web application as Figure 13, but with an error message displayed in a pink box: "Error: Usuario existente." (Error: User exists). The "Registrar usuario" section shows the "Usuario" field containing "admin" and the "Contraseña" field containing "\*\*\*\*\*". The "Registrar" button is still visible. The "Login" section remains unchanged with empty input fields and a "Login" button. The footer is the same as in Figure 13.

Figura 14: Creación de un usuario existente.

- Se puede acceder con las credenciales creadas:

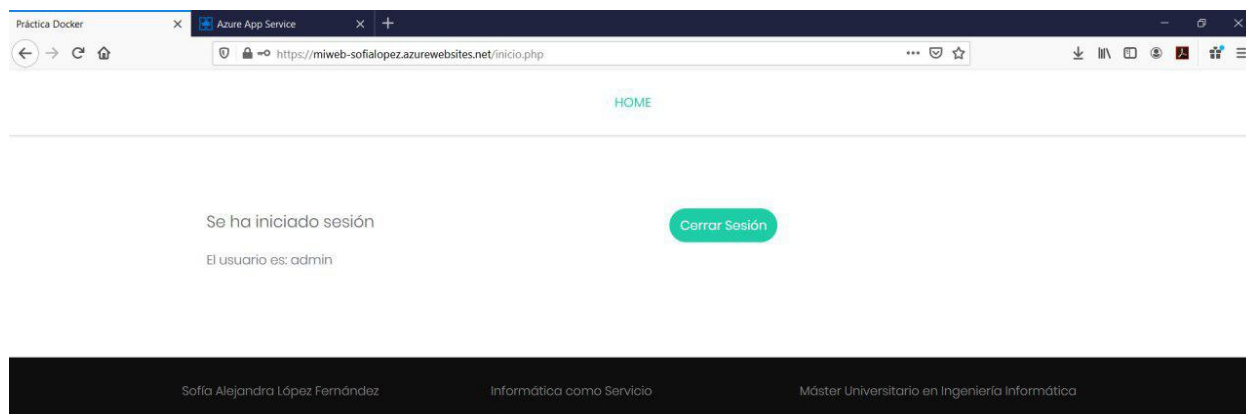


Figura 15: Inicio de sesión de un usuario.