Práctica de Azure

Iker García Calviño <iker.gcalvino@udc.es>

Despliegue de contenedor mediante Docker Compose en Azure

El objetivo de esta práctica es desplegar, a través de Docker Compose, un contenedor que contenga dos servicios con las imágenes de WordPress y MySQL. La comunicación entre los servicios se realiza de forma interna dentro del contenedor desplegado, exponiendo únicamente el puerto 80 del servicio WordPress al exterior.

Recursos necesarios en Azure

Azure App Service es el servicio utilizado para hospedar aplicaciones web, API REST y back-ends para dispositivos móviles. Puede ejecutar y escalar aplicaciones en entornos basados en Windows y Linux. Para lanzar un docker-compose en Azure App Service, es necesario desplegarlo sobre un Azure Web App.

Instrucciones

1. Generar un **Azure App Service** en Linux.
2. Una vez generado el **Web App**, es necesario configurar sus **App Settings** para indicar la ruta y las credenciales (usuario/contraseña) del **Azure Container Registry**. Esto permite que el servicio busque las imágenes en el registro de contenedores de Azure y no en DockerHub (por defecto).

Documentación

Descripción del despliegue

Se utilizaron las imágenes de WordPress y MySQL, descargadas desde DockerHub, para crear un contenedor mediante Docker Compose en Azure. A continuación, se detallan los pasos realizados:

1. **Descarga de imágenes desde DockerHub:**

   `docker pull wordpress:latest`

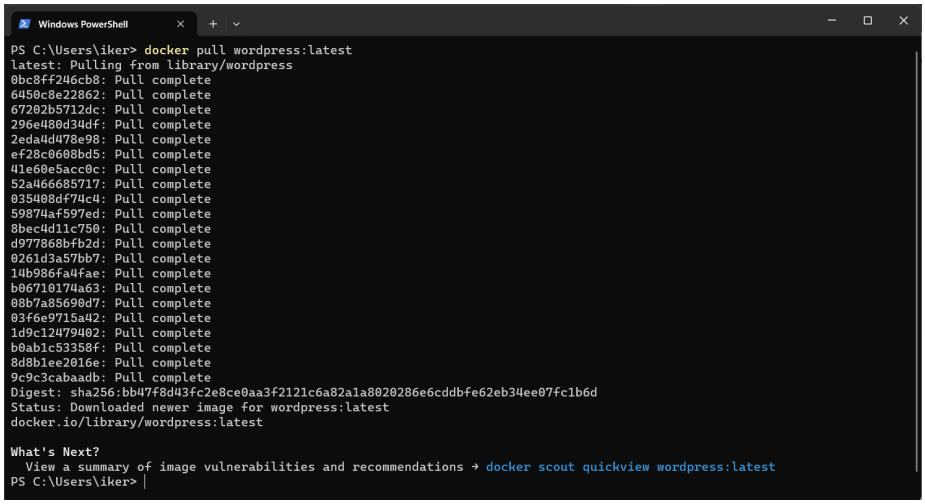

Figure 1: Descarga de la imagen de WordPress desde DockerHub

`docker pull mysql:5.7`

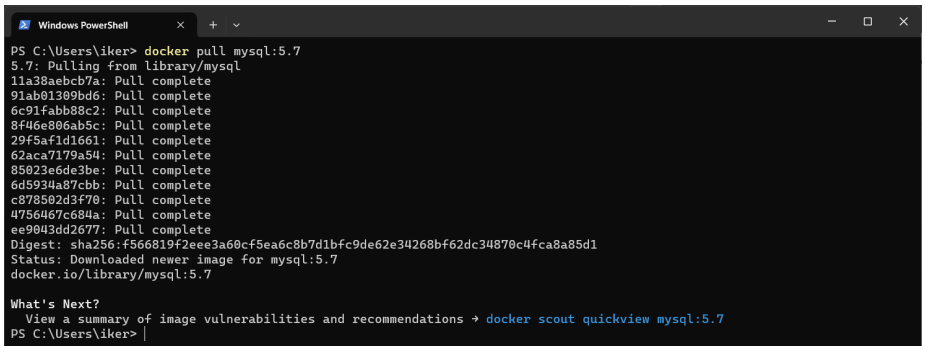

Figure 2: Descarga de la imagen de MySQL desde DockerHub

2. **Docker Compose:**

   Se creó un archivo `docker-compose.yml` para definir la configuración del servicio. Este archivo especifica los servicios, las imágenes, las variables de entorno y los volúmenes necesarios.

```yaml
version: '3'

volumes:
  db:
  wordpress:

services:
  db:
    image: mysql:5.7
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=wordpress
      - MYSQL_USER=test
      - MYSQL_PASSWORD=test
    ports:
      - "3306:3306"
    volumes:
      - db:/var/lib/mysql

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db:3306
      - WORDPRESS_DB_USER=test
      - WORDPRESS_DB_PASSWORD=test
      - WORDPRESS_DB_NAME=wordpress
    ports:
      - "8080:80"
    volumes:
      - wordpress:/var/www/html
```

   En este caso, se especifica que se van a desplegar 2 servicios: una base de datos (mysql:5.7) y una aplicación web (wordpress:latest). Este fichero también especifica que la aplicación web depende de la base de datos, por lo que se desplegará después de esta para evitar errores, y el uso de los puertos “3306:3306” y “8080:80”, respectivamente.

3. **Configuración en Azure:**

   - Creación de un grupo de recursos:

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

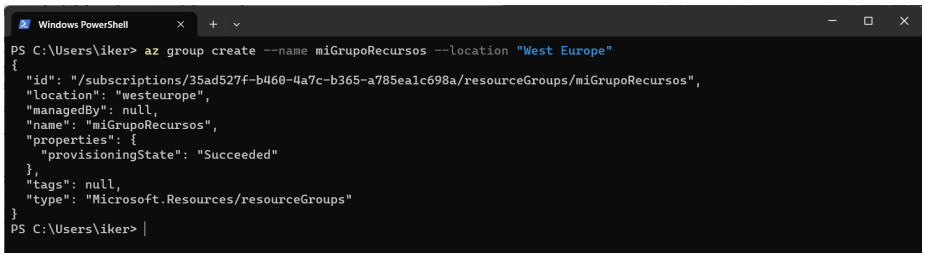

Figure 3: Creación de un grupo de recursos en Azure

- Creación de un registro de contenedores en Azure:

```
az acr create
--name ikergcalvinoregistry
--resource-group miGrupoRecursos
--sku Basic
```

Figure 4: Creación de un registro de contenedores en Azure

- Carga de imágenes en el sistema de registro de contenedores.

  Para que la aplicación web pueda utilizar las imágenes de docker, deberemos subirlas al registro de contenedores creado previamente.

  – Inicio de sesión en el registro de contenedores:

  ```
  az acr login
  --name ikergcalvinoregistry
  ```

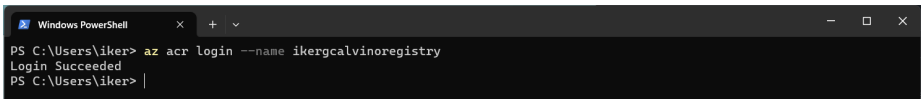

Figure 5: Inicio de sesión en el registro de contenedores de Azure

  – Tag y push de las imágenes al registro de contenedores de Azure:

  ```
  docker tag wordpress:latest ikergcalvinoregistry.azurecr.io/wordpress:latest
  docker tag mysql:5.7 ikergcalvinoregistry.azurecr.io/mysql:5.7
  ```

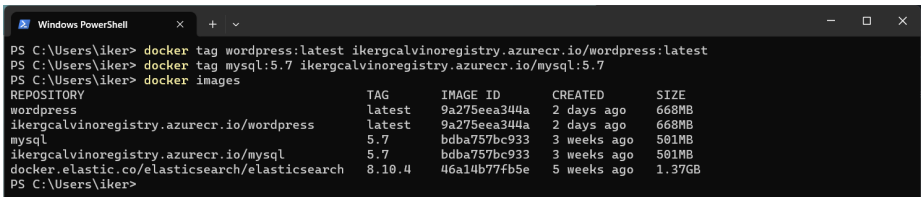

Figure 6: Etiquetado y envío de la imagen de WordPress al registro de contenedores de Azure

– Habilitación de la autenticación de administrador:

```
az acr update
-n ikergcalvinoregistry
--admin-enabled true
```

Figure 7: Habilitación de la autenticación de administrador en el registro de contenedores de Azure

– Inicio de sesión en el registro de contenedores de Azure y push de imágenes:

```
docker login ikergcalvinoregistry.azurecr.io
docker push ikergcalvinoregistry.azurecr.io/wordpress:latest
docker push ikergcalvinoregistry.azurecr.io/mysql:5.7
```

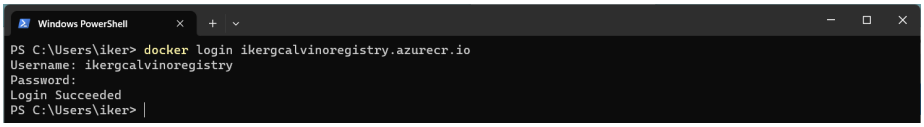

Figure 8: Inicio de sesión

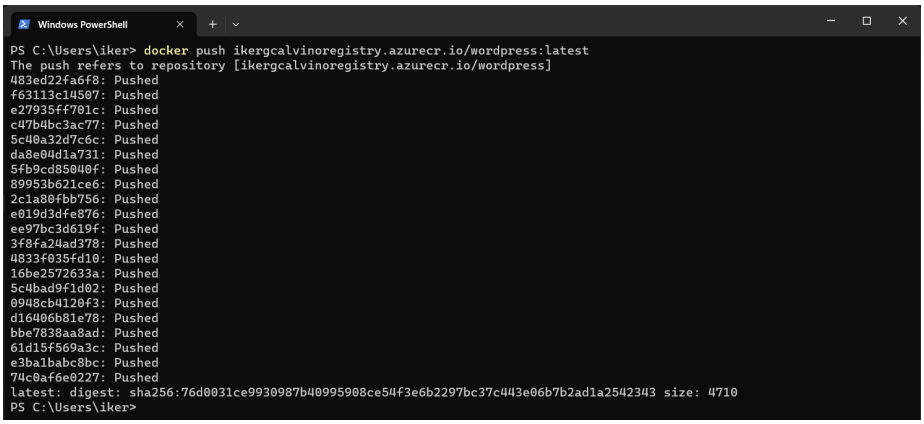

Figure 9: Envío de la imagen de WordPress al registro de contenedores de Azure

Figure 10: Envío de la imagen de MySQL al registro de contenedores de Azure

Podemos comprobar que se han subido correctamente desde la web de Azure.

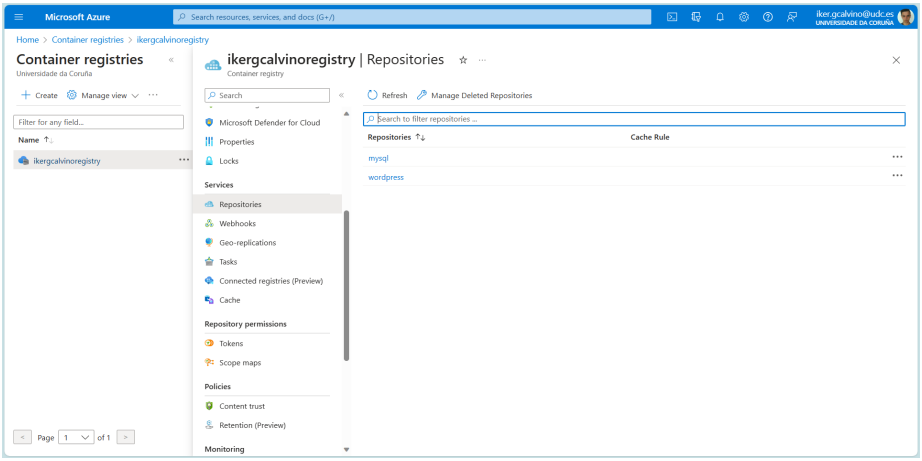

Figure 11: Verificación de imágenes en el portal de Azure

- Creación de un plan de servicio de aplicaciones:

```
az appservice plan create
--name ikergcalvino-appservice-plan
--resource-group miGrupoRecursos
--sku B1
--is-linux
```

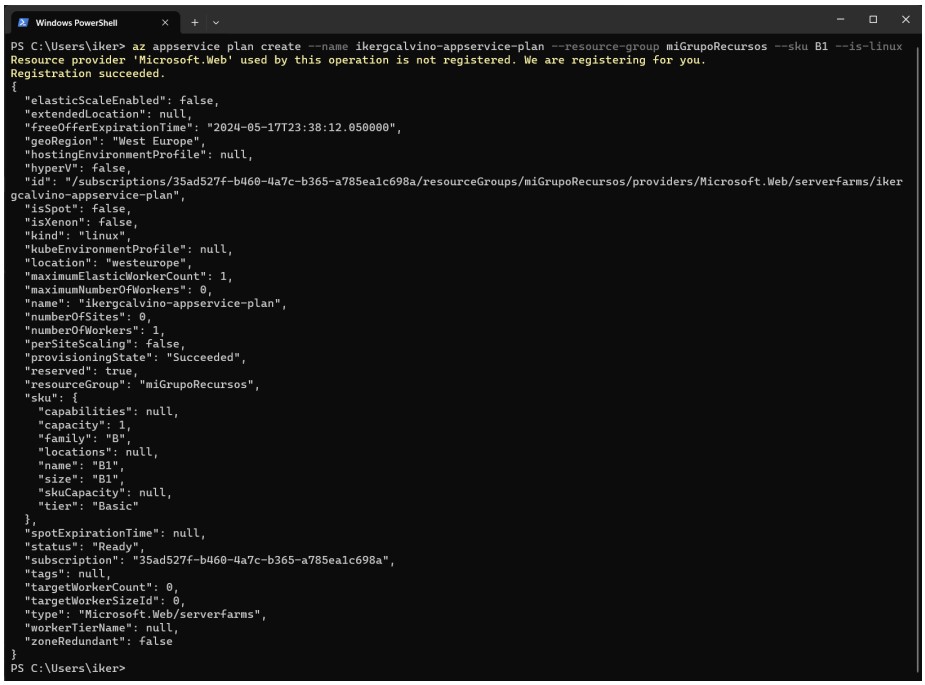

Figure 12: Creación de un plan de servicio de aplicaciones en Azure

- Creación de un servicio de aplicaciones web:

```
az webapp create
--name IkerGarciaWebApp
--plan ikergcalvino-appservice-plan
--resource-group miGrupoRecursos
--multicontainer-config-file docker-compose.yml
--multicontainer-config-type COMPOSE
```

Figure 13: Creación de un servicio de aplicaciones web en Azure (Parte 1)

Figure 14: Creación de un servicio de aplicaciones web en Azure (Parte 2)

– Configuración del registro de contenedores en el servicio de aplicaciones web:

  A continuación, configuramos la aplicación web para que acceda al registro de contenedores creado anteriormente.

```
az webapp config container set
--name IkerGarciaWebApp
--resource-group miGrupoRecursos
--docker-registry-server-url ikergcalvinoregistry.azurecr.io
--docker-registry-server-user ikergcalvinoregistry
--docker-registry-server-password nbXfPUWkGsKLypLyHcBdo8OWJUS6TA+4eJfZrs82n1+AC
```

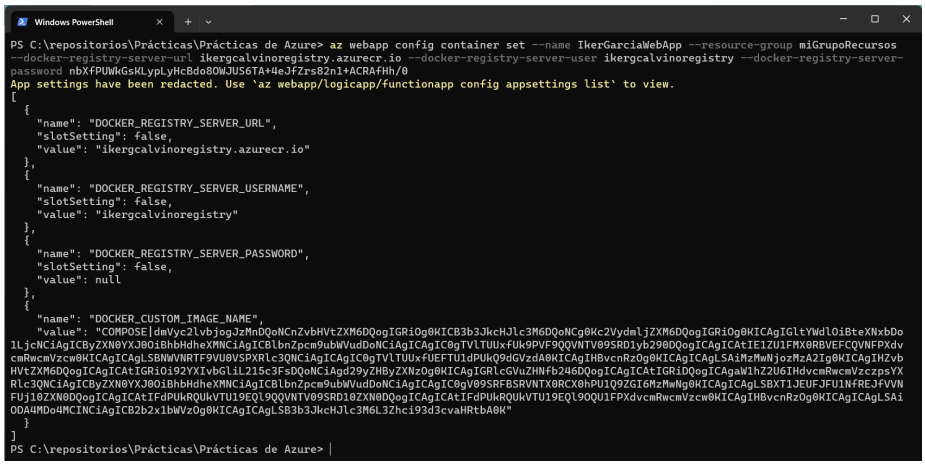

Figure 15: Configuración del registro de contenedores en el servicio de aplicaciones web de Azure

– Configuración adicional:

  Según la documentación oficial de Azure, se afirma que la persistencia de almacenamiento de datos está habilitada de forma predeterminada para las aplicaciones web en entornos Linux. Sin embargo, en una discusión de un hilo de Stack Overflow, se menciona lo contrario. Por ello, para evitar posibles problemas, ejecutaremos el siguiente comando:

```
az webapp config appsettings set
--resource-group miGrupoRecursos
--name IkerGarciaWebApp
--settings WEBSITES_ENABLE_APP_SERVICE_STORAGE=true
```

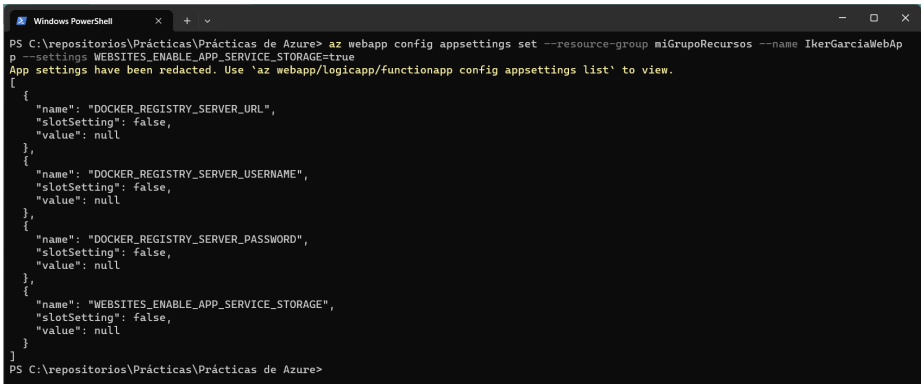

Figure 16: Configuración adicional en el servicio de aplicaciones web de Azure

4. **Acceso a la aplicación en Azure:**

   Al acceder por primera vez, es posible que necesites instalar y configurar WordPress. Para realizar esto, sigue estos pasos:

   1. Accede a la siguiente dirección en tu navegador:

   `https://ikergarciawebapp.azurewebsites.net/wp-admin/install.php`

   2. Selecciona el idioma de tu preferencia (en este caso, se ha escogido la opción de instalar WordPress en español).

   3. Completa la configuración requerida según tus preferencias. Puedes referirte a la documentación proporcionada en la práctica para obtener más detalles sobre la configuración realizada.

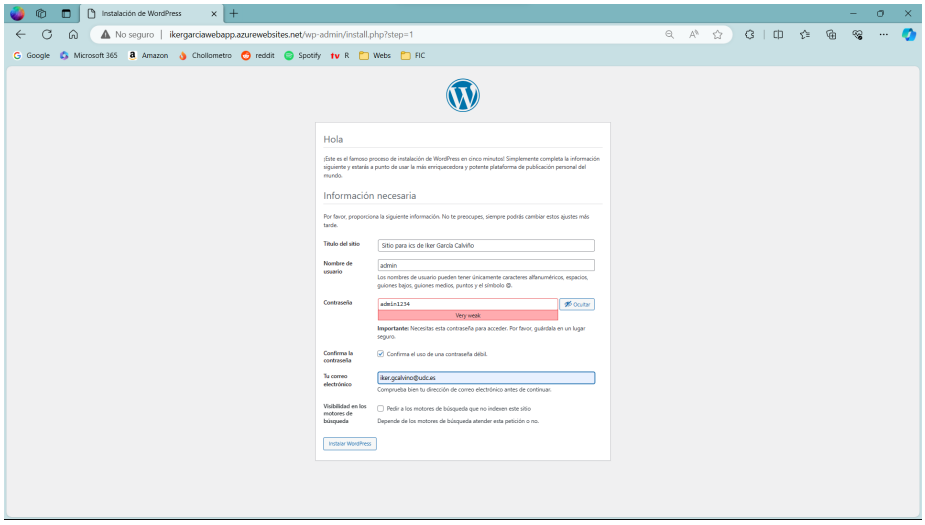

Figure 17: Acceso inicial a la instalación de WordPress en la aplicación web de Azure

4. Una vez completada la instalación, podrás observar la configuración en la interfaz de administración de WordPress.

Una vez completados los pasos anteriores, la aplicación estará disponible en la URL proporcionada por el servicio de aplicaciones web de Azure: IkerGarciaWebApp.

Figure 18: Sitio para ics de Iker García Calviño

Como vemos en la imagen, ya podemos tanto navegar por la web como crear, ver y comentar en los distintos posts que se encuentran en la web.