

# Yolo en Azure

Daniel Asensi Roch, dar33@alu.ua.es and Alexander Andonov Aracil,  
aaa@alu.ua.es

Universidad de Alicante, San Vicente del Raspeig, ESP

**Abstract.** Esta práctica proporciona una comprensión integral de la computación en la nube, específicamente a través del uso de contenedores en Azure Container Apps. Se enfoca en la creación y manejo de contenedores Docker, abarcando desde la escritura de scripts en Python y la preparación de Dockerfiles, hasta el despliegue de contenedores en la nube. Se ilustra con un ejemplo práctico que involucra un servidor web básico en Python, destacando los pasos necesarios para construir, ejecutar y desplegar un contenedor Docker. Además, se discuten aspectos como la gestión de imágenes de Docker y su integración con servicios de nube, proporcionando así una experiencia práctica en la implementación de aplicaciones basadas en contenedores en un entorno de nube.

**Keywords:** Computación en la Nube, Azure Container Apps, Docker, Contenedores, Python, Dockerfile, Despliegue en la Nube, Servidor Web, Gestión de Contenedores, Integración en la Nube

## 1 Introducción

En la era actual de la tecnología digital, la computación en la nube ha emergido como una solución fundamental para el almacenamiento, procesamiento y gestión de datos a gran escala. Esta práctica se centra en un aspecto crucial de la computación en la nube: el uso de contenedores, específicamente a través de Azure Container Apps y Docker. La capacidad de los contenedores para empaquetar y desplegar aplicaciones de manera eficiente y consistente en diferentes entornos es vital para los desarrolladores y administradores de sistemas. Esta sección introduce los conceptos básicos de los contenedores, Docker y Azure Container Apps, estableciendo una base para comprender su importancia y aplicabilidad en el campo de la computación en la nube.

### 1.1 Contexto

Los contenedores son una tecnología que permite encapsular una aplicación y su entorno de ejecución. A diferencia de las máquinas virtuales tradicionales, los contenedores comparten el mismo sistema operativo del host pero mantienen sus procesos aislados. Esto conduce a una mayor eficiencia y portabilidad, características esenciales en el desarrollo y despliegue de aplicaciones modernas.

Docker ha emergido como la plataforma de contenedores líder, proporcionando herramientas para desarrollar, desplegar y ejecutar aplicaciones dentro de contenedores. Ofrece una forma simplificada de manejar la dependencia y los problemas de configuración, asegurando que las aplicaciones funcionen de manera consistente en diferentes entornos.

Azure Container Apps es un servicio de Microsoft Azure que permite desplegar y escalar aplicaciones modernas basadas en contenedores. Integrado con el ecosistema de Azure, ofrece características como el escalado automático, la gestión de redes y la integración con otros servicios de Azure, lo que facilita la gestión de aplicaciones a gran escala en la nube.

## 2 Creación de recursos Hola Mundo

Esta sección aborda el proceso de creación y despliegue de una aplicación de contenedor simple, paso a paso. Comenzando con el desarrollo y prueba local de una aplicación "Hola Mundo", seguido por el despliegue de esta aplicación en un contenedor Docker y su subida a DockerHub, y finalmente, su implementación en Azure. Este proceso ilustra el flujo de trabajo completo desde el desarrollo hasta el despliegue en la nube.

### 2.1 Prueba Hola Mundo local

El primer paso en el desarrollo de aplicaciones de contenedores es la creación y prueba local. Aquí, se desarrolla una aplicación básica "Hola Mundo" utilizando Python. Se explica cómo escribir un script simple que imprime "Hola Mundo" y cómo ejecutar este script localmente para asegurarse de que funciona según lo esperado. Este paso es esencial para validar la funcionalidad básica antes de proceder a la contenerización.

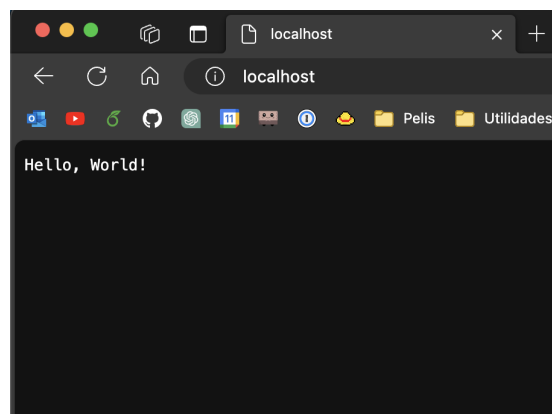


Fig. 1.

## 2.2 Contenedor subido a DockerHub

Una vez que la aplicación "Hola Mundo" se ha probado localmente, el siguiente paso es contenerizarla usando Docker. En esta subsección, se describe cómo crear un Dockerfile, que especifica cómo se debe construir la imagen del contenedor para la aplicación. Luego, se detalla el proceso de construir la imagen de Docker localmente, probar el contenedor para asegurarse de que se ejecuta correctamente, y finalmente, subir esta imagen a DockerHub, un registro de contenedores en la nube que permite almacenar y compartir imágenes de Docker.

```
> docker build -t hello-az-server .
[*] Building 19.7s (4/7)                                docker:desktop-linux
=> [internal] load build definition from dockerfile      0.0s
=> => transferring dockerfile: 398B                     0.0s
=> [internal] load .dockerignore                        0.0s
=> => transferring context: 2B                           0.0s
=> [internal] load metadata for docker.io/library/python:3.8-slim 2.9s
=> [1/5] FROM docker.io/library/python:3.8-slimsha256:3cb3a8d4fa08f89921c9e788618c51 16.7s
=> => resolve docker.io/library/python:3.8-slimsha256:3cb3a8d4fa08f89921c9e788618c51 0.0s
=> => sha256:50f30729fa3ef47986ae433e765706c2e85fa0fcf79cd3c8105ad9c9 6.95kB / 6.95kB 0.0s
=> => sha256:1f7ce2fa46ab3942feabee654933948821303a5a821789dddb2d8 5.24MB / 29.15MB 16.7s
=> => sha256:402c5d63eaf9e967557fc9e991913677ca8d5112a8e631b16677 1.05MB / 3.32MB 16.7s
=> => sha256:b5b6aaef47c7fa37759532a8864ab919a3b5a9687ef9889a8d27c 2.18MB / 13.72MB 16.7s
=> => sha256:3cb3a8d4fa08f89921c9e788618c515a7cbb5f0e0c531dc9b657cf9f 4.42kB / 4.42kB 0.0s
=> => sha256:10425670f3dc20aaee44b36797109183636cb6896ccded30377a3e2 1.25kB / 1.25kB 0.0s
=> [internal] load build context                        0.0s
=> => transferring context: 316.29kB                     0.0s
```

Fig. 2.




<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	magical_shamir c7e8cf2ee73c	hello-az-server	Running	0.23%	80.80	46 seconds ago	  

Fig. 3.

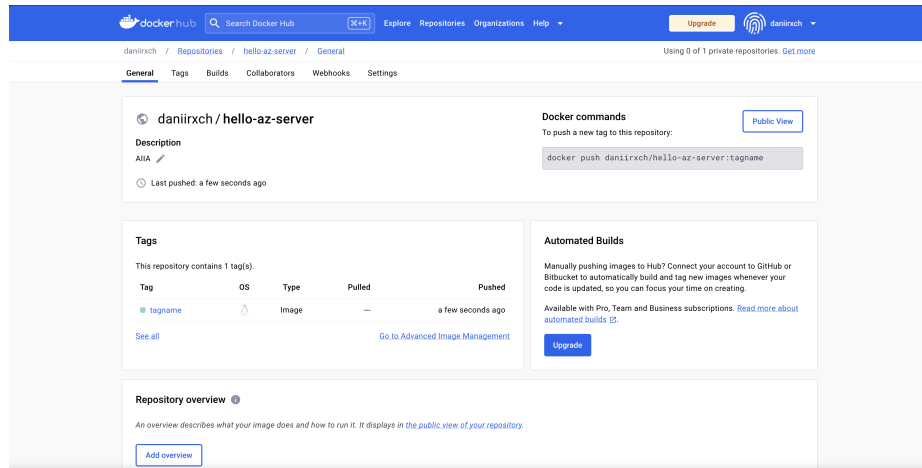


Fig. 4.

## 2.3 Creación del Hola Mundo en Azure

El último paso es desplegar la aplicación "Hola Mundo" en la nube, utilizando Azure Container Apps. Esta subsección guía a través del proceso de configuración de un entorno en Azure, incluyendo la creación de una cuenta, la configuración de un grupo de recursos, y el despliegue de la aplicación utilizando la imagen de Docker almacenada en DockerHub. También se cubren aspectos como la configuración de la red y la seguridad, y cómo acceder a la aplicación una vez que está en funcionamiento en Azure.

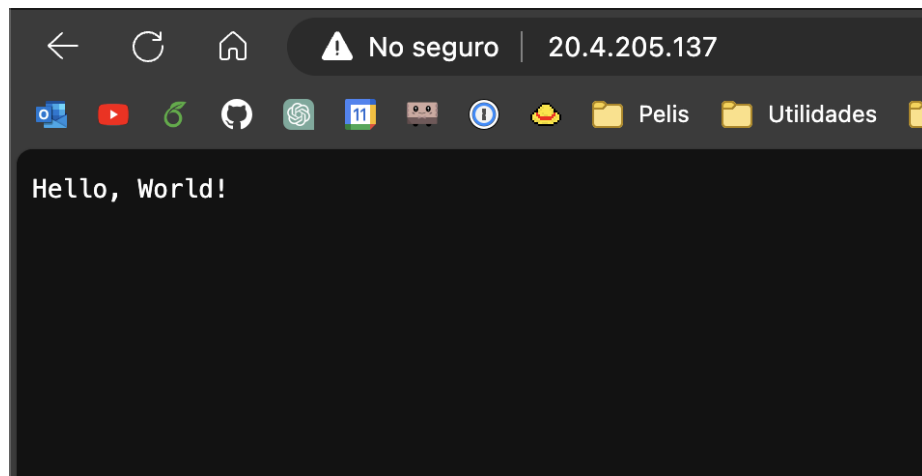


Fig. 5.

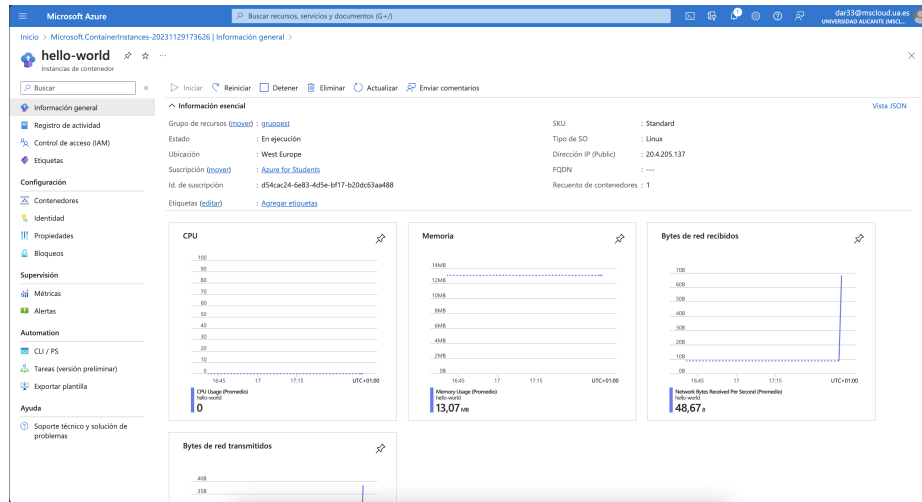


Fig. 6.

### 3 Contenedor de Yolo

Esta sección se centra en el desarrollo y despliegue de un contenedor Docker que ejecuta YOLO, un algoritmo de detección de objetos avanzado. Se detallan los pasos desde la configuración del Dockerfile, la subida de la imagen a DockerHub, hasta la creación y prueba del recurso en un entorno de nube.

#### 3.1 Dockerfile configurado

El primer paso en la creación del contenedor de Yolo es configurar el Dockerfile. Se explica cómo preparar un Dockerfile que instala todas las dependencias necesarias, incluyendo Python, las bibliotecas necesarias, y el propio modelo YOLO. Se aborda también cómo estructurar el Dockerfile para optimizar la construcción de la imagen y asegurar su funcionamiento eficiente.

1. **Base de la Imagen:** Utiliza `httpd:latest` como imagen base, que es el servidor web Apache HTTP Server.
2. **Instalación de Dependencias:** Instala Python, pip (gestor de paquetes de Python), un entorno virtual de Python y OpenCV para Python, necesarios para ejecutar el script de detección de objetos.
3. **Configuración del Entorno Virtual Python:** Crea y configura un entorno virtual en `/opt/venv` para manejar las dependencias de Python de forma aislada.
4. **Instalación de Paquetes Python:** Instala `yolov5`, `numpy`, y `opencv-python` para el procesamiento de imágenes y la ejecución del modelo YOLO.

5. **Copia de Archivos:** Transfiere archivos esenciales al contenedor, incluyendo scripts de Python y JavaScript, el modelo de YOLO (yolov8n.pt), y archivos HTML para la interfaz del servidor web.
6. **Permisos de Archivos:** Otorga permisos de ejecución a los scripts grabber.py y start.sh.
7. **Exposición de Puertos y Comando de Inicio:** Expone el puerto 80 y especifica que el script start.sh se ejecute al iniciar el contenedor, el cual debe iniciar tanto el servidor Apache como el script de Python.

```
#!/bin/bash
# Iniciar Apache en segundo plano
apachectl start
# Ejecutar el script de Python
cd /usr/local/apache2/htdocs/

exec python3 grabber.py

FROM httpd:latest

RUN apt-get update && apt-get install -y python3 python3-pip python3-venv python3-opencv
RUN python3 -m venv /opt/venv
ENV PATH="/opt/venv/bin:$PATH"

RUN pip install yolov5 numpy opencv-python

COPY index.html /usr/local/apache2/htdocs
COPY not_found.html /usr/local/apache2/htdocs
COPY script.js /usr/local/apache2/htdocs
COPY yolov8n.pt /usr/local/apache2/htdocs
COPY grabber.py /usr/local/apache2/htdocs
COPY start.sh /usr/local/apache2/htdocs

RUN chmod 777 /usr/local/apache2/htdocs/grabber.py
RUN chmod +x /usr/local/apache2/htdocs/start.sh

EXPOSE 80

CMD ["/usr/local/apache2/htdocs/start.sh"]
```

**Subida a Dockerhub** Una vez que el contenedor de Yolo se ha configurado y probado localmente, se procede a subir la imagen a DockerHub. En esta subsección, se describe el proceso de etiquetado y subida de la imagen, incluyendo cómo configurar las etiquetas para una organización y versiones específicas. Se

proporcionan detalles sobre cómo manejar las credenciales de DockerHub y asegurar que la imagen esté disponible para su despliegue.

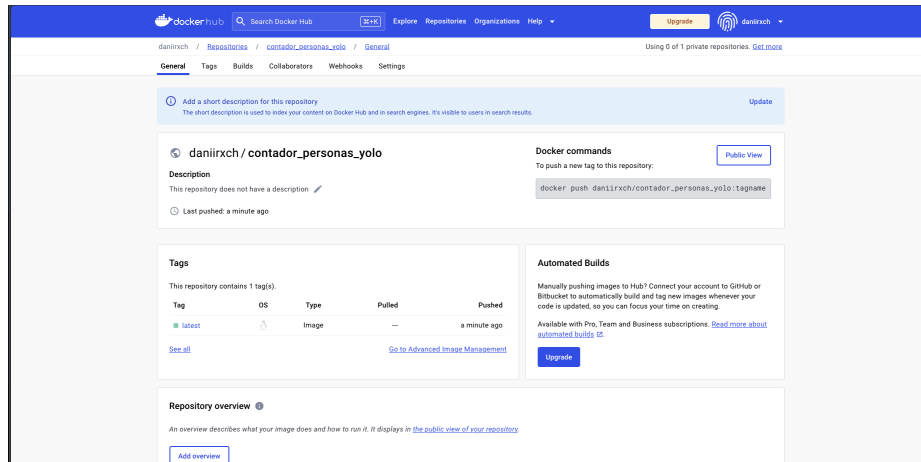


Fig. 7.

### 3.2 Creación del recurso

El siguiente paso es la creación del recurso en la nube. Se detalla cómo utilizar servicios de nube, como Azure Container Instances o Azure Kubernetes Service, para desplegar el contenedor de Yolo. Se cubren aspectos como la configuración del entorno, la asignación de recursos (CPU, memoria), y la configuración de redes y seguridad para el contenedor.

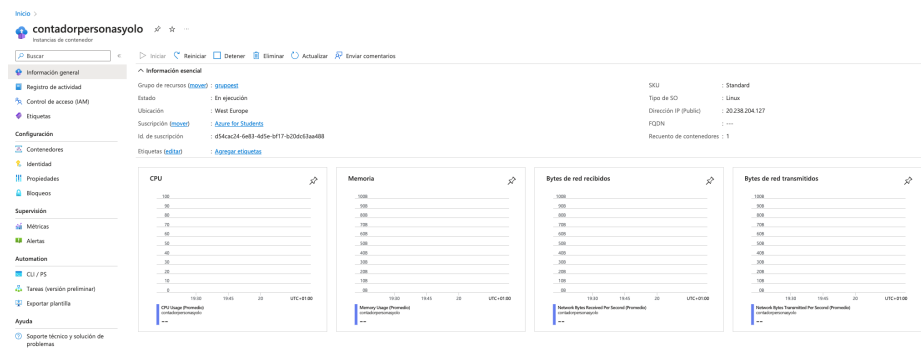


Fig. 8.

### 3.3 Prueba

Finalmente, se describe cómo realizar pruebas para asegurarse de que el contenedor de Yolo está funcionando correctamente en la nube. Se incluyen pasos para acceder al contenedor, monitorear su rendimiento y verificar la precisión y eficacia del modelo YOLO en la detección de objetos. Se pueden ofrecer ejemplos de pruebas y métricas para evaluar el funcionamiento del contenedor.

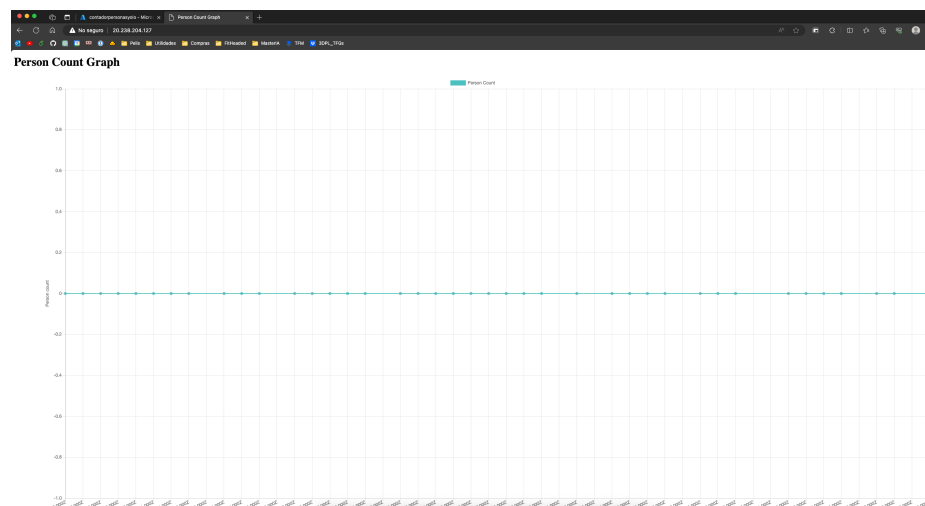


Fig. 9.



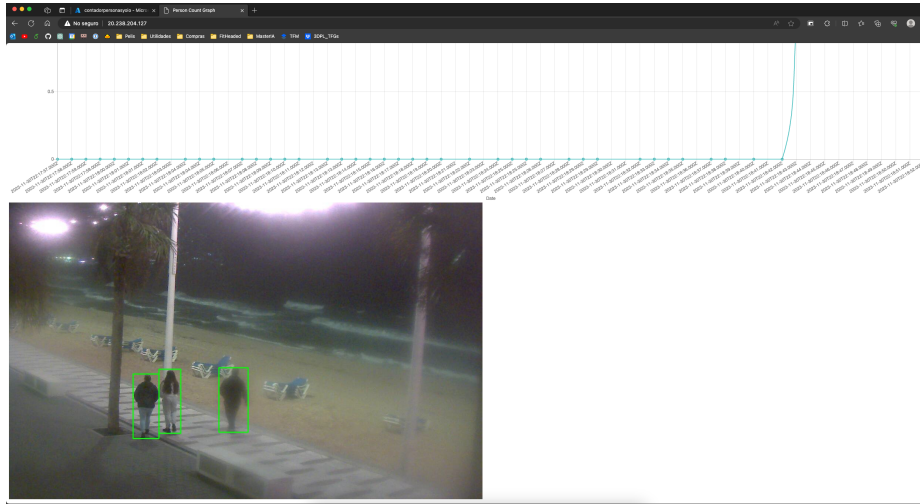


Fig. 10.

## 4 Opinión Personal

El uso de Docker ha revolucionado la forma en que se desarrollan, prueban y despliegan aplicaciones. En mi experiencia, la capacidad de Docker para empaquetar aplicaciones y sus dependencias en contenedores autónomos simplifica enormemente el proceso de desarrollo. Esta contenerización asegura que las aplicaciones funcionen de manera consistente en diferentes entornos, eliminando el clásico problema de "en mi máquina sí funciona".

Además, la integración con servicios en la nube, como Azure Container Apps, potencia aún más las capacidades de Docker. La facilidad de despliegue y la escalabilidad que ofrecen estos servicios en la nube son invaluable, especialmente en aplicaciones que requieren alta disponibilidad y flexibilidad para manejar variaciones en la carga de trabajo.

En particular, la implementación de la tecnología YOLO en un contenedor Docker es un ejemplo fascinante de cómo las tecnologías avanzadas de inteligencia artificial pueden integrarse en aplicaciones web. Esta experiencia ha sido reveladora en términos de las posibilidades que abre para el desarrollo de aplicaciones inteligentes y receptivas. La detección de objetos en tiempo real tiene innumerables aplicaciones, desde la seguridad hasta el análisis de datos en tiempo real, y poder implementar esto en un entorno de contenedor es impresionante.