ARQUITECTURAS E INFRAESTRUCTURAS PARA INTELIGENCIA ARTIFICIAL
PRÁCTICA 4: COMPUTACIÓN EN LA NUBE – SEGUNDA PARTE

Objetivos

- Conocer las posibilidades de la computación en la nube.

Trabajo Guiado:

USO DE CONTENEDORES CON AZURE CONTAINER APPS

Los contenedores son un tipo de virtualización ligera basada en el sistema operativo. Componen la aplicación a ejecutar junto a las bibliotecas y herramientas del sistema operativo necesarias. A continuación, se van a explicar los pasos necesarios para crear un contenedor que ejecuta un script de Python que escucha en el puerto de red 80.

Se va a usar la plataforma **Docker** (https://www.docker.com/) para crear, desplegar, ejecutar y gestionar los contenedores. Puedes ampliar conocimientos sobre esta tecnología en: https://docs.docker.com/get-started/

1) Crear script en código Python

Crea una carpeta en tu máquina local y guarda dentro el siguiente script de Python como "hello.py":

```
import http.server
import socketserver

# Define the request handler
class HelloHandler(http.server.BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/plain')
        self.end_headers()
        self.wfile.write(b'Hello, World!')

# Set up the server
with socketserver.TCPServer(('', 80), HelloHandler) as httpd:
    print('Server started on port 80...')
    httpd.serve_forever()
```

Este script responde "Hello, World!" a las peticiones que recibe. Este ejemplo utiliza el protocolo http, que es de más alto nivel que el uso de sockets, y por lo tanto, más sencillo de utilizar.

Si se quiere comprobar el script, abre una terminal y ejecútalo con "sudo python hello.py". Al utilizar el puerto 80 se necesitan permisos de administrador, por eso se ejecuta como "sudo". En el navegador escribe la URL "localhost". Debería aparecer el texto "Hello, World!".

2) Crear Dockerfile

El archivo Dockerfile se utiliza para especificar como se crea una imagen de Docker.

Crea un fichero dentro de la carpeta anterior llamado “Dockerfile”, y copia el siguiente contenido:

```
# Use an official Python runtime as a parent image
FROM python:3.8-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Make port 80 available to the world outside this container
EXPOSE 80

# Run hello.py when the container launches
CMD ["python", "hello.py"]
```

3) Crear la imagen del contenedor

Ejecuta el comando Docker para crear la imagen a partir del Dockerfile:

```
>docker build -t hello-az-server
```

La imagen del contenedor debería de aparecer en la lista de imágenes de Docker:

```
>docker image ls
```

Para eliminar la imagen:

```
>docker image rm hello-az-server
```

4) Ejecución del contenedor

En primer lugar, ejecuta el contenedor en local para comprobar que se ha creado correctamente:

```
>docker run –rm -p 80:80 hello-az-server
```

Es importante entender que un contenedor docker tiene el ciclo de vida de la aplicación que se está ejecutando. En el fichero Dockerfile, la aplicación se define con el comando de entrada (CMD). Esto significa que, si la aplicación termina, el contenedor también se parará.

A continuación, comprueba si funciona en el navegador. Ejecuta “localhost” y comprueba si funciona.

En caso de error, se puede acceder a los logs del contenedor con:

```
>docker logs hello-az-server
```

5) Subir el contenedor al Hub de Docker para alojar la aplicación

Crea una cuenta en DockerHub (https://hub.docker.com/).

DockerHub es un registro para repositorios de software en la nube, es decir, actúa como una biblioteca para las imágenes Docker. Es un servicio en línea que contiene repositorios públicos y privados.

Una vez creada la cuenta, crea un repositorio, haciendo click en “Create repository”.

- Repository name: hello-az-server
- Redacta una breve descripción
- Mantén la visibilidad pública

Sube la imagen creada al repositorio de Docker Hub mediante los siguientes pasos:

- Agrega una etiqueta “tag” a la imagen, utilizando la nomenclatura nombreusuario/nombrerepositorio:

```
>docker tag hello-az-server nombreusuario/hello-az-server
```

- Identifícate en el sistema:

```
>docker login
```

- Sube la imagen al repositorio indicando el tag asignado:

```
>docker push nombreusuario/hello-az-server
```

Puedes comprobar que se ha subido correctamente entrando en tu cuenta de DockerHub.

6) Despliegue del contenedor en MS Azure

Inicia sesión en MS Azure y busca el servicio “Aplicaciones Contenedoras”
Haz click en “Crear”

Detalles del proyecto:

- Subscripción: Azure para estudiantes
- Grupo de recursos: grupoest
- Nombre de la aplicación: hello-world
- Región: West europe
- Entorno de container apps: Dejar por defecto

- Click en “Siguiente: Contenedor”
- Deseleccionamos “Usar imagen de inicio rápido”
- Nombre: hello-world
- Origen de imagen: Docker Hub u otros registros
- Tipo de imagen: Público
- Servidor de inicio de sesión de registro: docker.io
- Imagen y etiqueta: nombreusuario/hello-az-server:latest
- Invalidación de comando: Dejar en blanco
- CPU y memoria: Poner la opción más baja

- Click en "Revisar y crear"
- Click en "Crear"

Por último, hay que habilitar el acceso a la aplicación desde una IP pública.

- Accede al recurso recién creado. Click en "Ir al recurso" si sigues en la pantalla anterior.
- En el menú lateral izquierdo, selecciona "Entrada"
- Marca el checkbox "Entrada" para habilitarla. Aparecerán más opciones.
- Tráfico de entrada: Aceptando tráfico desde cualquier lugar
- Puerto de destino: 80

Para probar la aplicación, vamos a "Información general" en el menú lateral izquierdo, y la URL la encontramos en "Dirección URL de la aplicación". Introducimos la URL en el navegador y probamos que la aplicación funciona correctamente.

Trabajo:

Ejecuta la aplicación de la práctica 3 en Azure Container Apps. Para ello deberás crear un contenedor con la aplicación, subirlo a Dockerhub, y lanzar una aplicación contenedora en Azure con ese contenedor.

Para crear el contenedor se recomienda partir de la imagen básica de "ubuntu:latest", y añadir los paquetes necesarios utilizando la instrucción RUN:

```
>RUN apt install …
>RUN pip install …
```

En los contenedores docker no se suelen utilizar gestores de servicios como *systemd*, ya que estos son pesados en cuanto a los recursos que utilizan. Para ejecutar el servidor web y el script de Python la opción más sencilla es crear un "script bash" que lance las dos aplicaciones, y utilizar ese script como punto de entrada del contenedor mediante el siguiente código:

```
>#!/bin/sh
>apachectl start
>python3 grabber.py
```

Redacta una memoria de prácticas en la que describas los pasos seguidos, los problemas encontrados y las decisiones de diseño para la resolución del trabajo a realizar. Añade un apartado de conclusiones del trabajo realizado. Incluye las capturas de pantallas que veas necesarias para demostrar que has seguido los pasos indicados en el enunciado.

Normas de entrega:

- La realización del trabajo es por parejas.
- El documento debe seguir el formato definido para las publicaciones de *Lecture Notes in Computer Science* de *Springer*
  https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines
- La entrega se realizará a través de un adjunto a una tutoría de campus virtual.
- Los formatos válidos del documento son *MS Word* (.doc, .docx), *OpenDocument* (.odt) o *Portable Document Format* (.pdf).