CUCEI

ACTIVIDAD 08

K8S-EJEMPLO

MATERIA:
Computación tolerante a fallas (I7036)
NRC:
179961
SECCIÓN:
D06
MAESTRO:
López Franco, Michel Emmanuel
HORARIO:
Lunes y Miércoles, 11:00 – 12:55
FECHA DE ENTREGA:
27/04/2023
ALUMNO:
J. Emmanuel Ortiz Renteria (219747611)

Contenido

Introducción

Kubernetes es una plataforma de código abierto diseñada para automatizar, escalar y gestionar aplicaciones contenerizadas. Kubernetes y Docker funcionan como tecnologías complementarias. Docker se utiliza para crear y empaquetar contenedores, mientras que Kubernetes se encarga de orquestar y gestionar esos contenedores en un entorno distribuido. Kubernetes puede gestionar contenedores creados con Docker y otras herramientas de contenerización, pero Docker es una de las opciones más populares y ampliamente utilizadas.

Usos de Kubernetes:

1. **Orquestación de contenedores:** Kubernetes simplifica la gestión de aplicaciones contenerizadas, permitiendo escalar automáticamente, distribuir y gestionar el estado de los contenedores en un clúster.
2. **Despliegue de aplicaciones:** Facilita la implementación y actualización de aplicaciones de forma rápida y consistente, lo que acelera el ciclo de desarrollo y reduce el tiempo de inactividad.
3. **Escalabilidad:** Permite escalar horizontalmente aplicaciones según la demanda del tráfico, distribuyendo la carga de manera equitativa entre los nodos del clúster.
4. **Alta disponibilidad:** Kubernetes proporciona herramientas para garantizar la alta disponibilidad de las aplicaciones, mediante la detección automática de fallos y la recuperación automática.

Ventajas de Kubernetes:

- **Portabilidad:** Permite ejecutar aplicaciones de manera consistente en diferentes entornos, ya sea en la nube pública, privada o en entornos locales.
- **Escalabilidad:** Facilita la escalabilidad de las aplicaciones, permitiendo añadir o quitar recursos de forma dinámica según las necesidades del negocio.
- **Automatización:** Kubernetes automatiza tareas como el aprovisionamiento de recursos, la gestión del almacenamiento y la distribución de carga, lo que reduce la carga operativa y los errores humanos.

Desventajas de Kubernetes:

- **Curva de aprendizaje:** Puede requerir tiempo y esfuerzo para familiarizarse con los conceptos y la terminología de Kubernetes, lo que puede dificultar su adopción inicial.
- **Complejidad:** La gestión de un clúster de Kubernetes puede ser compleja, especialmente en entornos de producción, donde se requiere configuración avanzada y mantenimiento continuo.

- **Recursos:** Kubernetes puede consumir recursos significativos, tanto en términos de hardware como de recursos humanos, lo que puede ser una limitación para algunas organizaciones.

Explicación del programa

Usando Docker se crea una imagen, que instala temporalmente el modulo **univorn** y **fastapi** en el contenedor para luego correr “main.py”, después esa imagen es empujado “push” hacia un repertorio de Docker hub y luego descargado “push” para volverse una imagen para un deployment y será ejecutado en un cluster que correra con configuración de “deployment.yaml” . El programa es simple, se crea una identidad formado de un numero id, un nombre completo (primer nombre, 20% de probabilidad de agregar un segundo nombre, dos apellidos), una edad (entre 18 y 45 años) y una fecha de nacimiento que corresponde a esa edad.

Requerimientos

- Python 3+ Verifica que Python esté disponible en tu PATH para que puedas ejecutarlo desde la línea de comandos.
- DockerCLI. Verifica que Docker esté disponible en tu PATH para que puedas ejecutarlo desde la línea de comandos.
- Kuberctl: Ayuda a manejar la creación de pods/deployments.
- Minikube: Entorno para crear clusters de manera local.
- Visual Studio Code.
- Dockerfile: El proyecto debe contener un Dockerfile que incluya todas las dependencias necesarias, incluyendo Python y el módulo **requests**.
- Deployments.yaml: Un archivo con todas las descripciones para crear un pod/deployment para ejecutar en el cluster para kuebrnetes.
- Sistema Operativo:
    - Windows 10 o superior
    - Linux
    - Mac Os

Archivos y códigos:

__init__.py:

Solo se crea un archivo vacío, al correr la dependencia Python en la imagen se cargara los recursos necesarios a este.

Código de main.py:

```python
import tkinter as tk
from datetime import date, timedelta
import random
from fastapi import FastAPI

app = FastAPI()

class identidad():
    nombre_completo = str("")
    edad = int(0)
    nid = int()
    fecha_de_nacimiento = str()


def descargar_assets():
    nombres = "Hugo Mateo Martín Lucas Leo Daniel Alejandro Manuel Pablo
Álvaro Adrián Enzo Mario Diego David Oliver Marcos Thiago Marco Álex
Javier Izan Bruno Miguel Antonio Gonzalo Liam Gael Marc Carlos Juan Ángel
Dylan Nicolás José Sergio Gabriel Luca Jorge Darío Íker Samuel Eric Adam
Héctor Francisco Rodrigo Jesús Erik Amir Jaime Ian Rubén Aarón Iván Pau
Víctor Guillermo Luis Mohamed Pedro Julen Unai Rafael Santiago Saúl
Alberto Noah Aitor Joel Nil Jan Pol Raúl Matías Martí Fernando Andrés
Rayan Alonso Ismael Asier Biel Ander Aleix Axel Alan Ignacio Fabio Neizan
Jon Teo Isaac Arnau Luka Max Imran Youssef Anas Elías"
    apellidos = "Hernández García Martínez López González Pérez Rodríguez
Sánchez Ramírez Cruz Gómez Flores Morales Vázquez Jiménez Reyes Díaz
Torres Gutiérrez Ruiz Mendoza Aguilar Méndez Moreno Ortíz Juárez Castillo
Álvarez Romero Ramos Rivera Chávez De_la_Cruz Domínguez Guzmán Velázquez
Santiago Herrera Castro Vargas Medina Rojas Muñóz Luna Contreras Bautista
Salazar Ortega Guerrero Estrada"
    #Descodificar el contenido
    lista_nombres = nombres.split()
    lista_apellidos = apellidos.split()
    return lista_nombres,lista_apellidos

def generar_id_random():
    n_id = random.randint(1,1000)
    return n_id

def edad_random():
    return random.randint(18,45)

def fecha_random(edad):
    hoy = date.today()
    margen_anyo = int(hoy.year) - edad
    fin = date(margen_anyo,int(hoy.month),int(hoy.day))
    inicio = fin - timedelta(days=365)
    fecha = inicio + (fin - inicio) * random.random()
    fecha = str(fecha)
    return fecha
```

```
def generar_nombre():
    lista_nombres,lista_apellidos = descargar_assets()

    # Generar el nombre
    nombre_completo = random.choice(lista_nombres)
    if((random.randint(1,100) <= 20)):
        nombre_completo = nombre_completo +" "+
str(random.choice(lista_nombres))
    nombre_completo = nombre_completo +" "+
random.choice(lista_apellidos)
    nombre_completo = nombre_completo +" "+
random.choice(lista_apellidos)
    nombre_completo = nombre_completo.replace("_"," ")
    return nombre_completo

def generar_identidad():
    id_tmpr = identidad()
    id_tmpr.nid = generar_id_random()
    id_tmpr.nombre_completo = generar_nombre()
    id_tmpr.edad = edad_random()
    id_tmpr.fecha_de_nacimiento = fecha_random(id_tmpr.edad)
    return id_tmpr

@app.get("/")
def read_root():
    try:
        id_contenedor = generar_identidad()
        generar_identidad()
        return f"Mi nombre es {id_contenedor.nombre_completo}, tengo
{id_contenedor.edad} años, naci el {id_contenedor.fecha_de_nacimiento}.
Mi ID es {id_contenedor.nid}."
    except Exception as e:
        return "Ocurrio el error:", e.args
```

Código de Dockerfile:

```
FROM python
WORKDIR /code
COPY ./requirements.txt /code/requirements.txt
RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt
COPY ./app /code/app
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "80"]
```

Código de requirements.txt:

```
requests~=2.31.0
fastapi~=0.110
uvicorn~=0.29
```

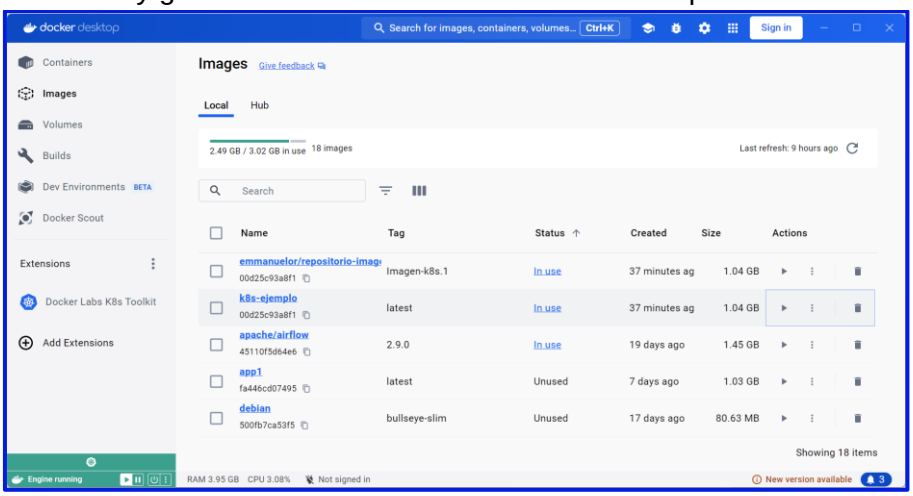
Código de deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: appk
  labels:
    app: appk
spec:
```

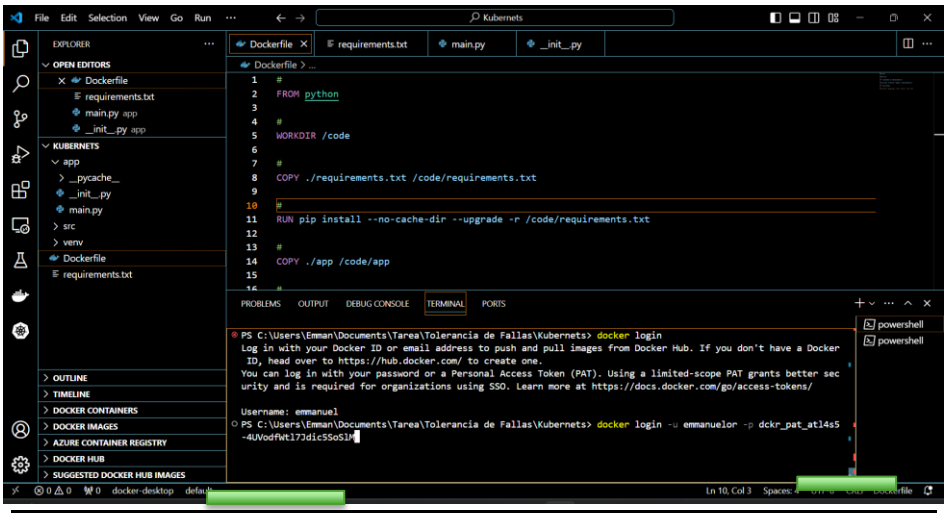
```yaml
  replicas: 1
  selector:
    matchLabels:
      app: appk
  template:
    metadata:
      labels:
        app: appk
    spec:
      containers:
      - name: app
        image: k8s-ejemplo:latest
        ports:
        - containerPort: 80
        volumeMounts:
        - mountPath: /tmp/data
          name: test-volume
      volumes:
        - name: test-volume
          hostPath:
            path: /tmp/data
---
apiVersion: v1
kind: Service
metadata:
  name: appk
spec:
  type: NodePort
  selector:
    app: appk
  ports:
  - name: appk-port
    protocol: TCP
    port: 8080
    targetPort: 8080
```

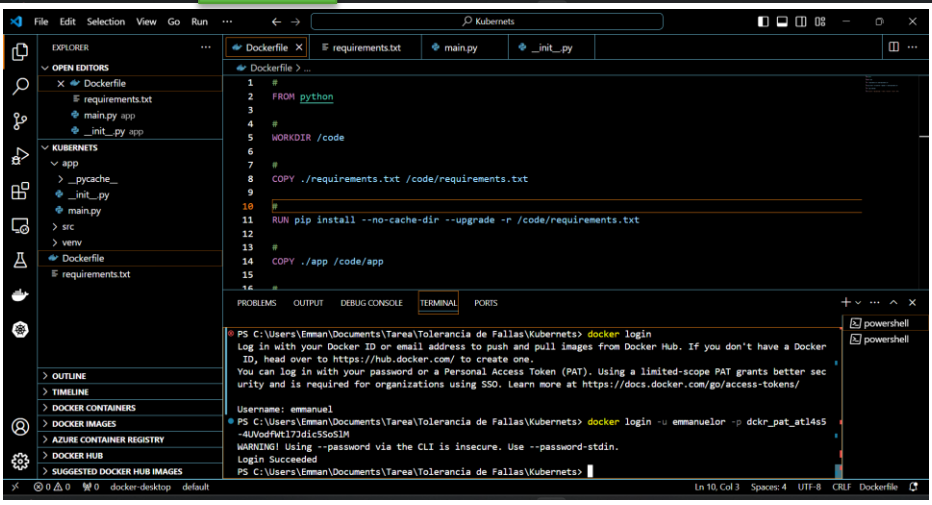
Capturas de ejecución:

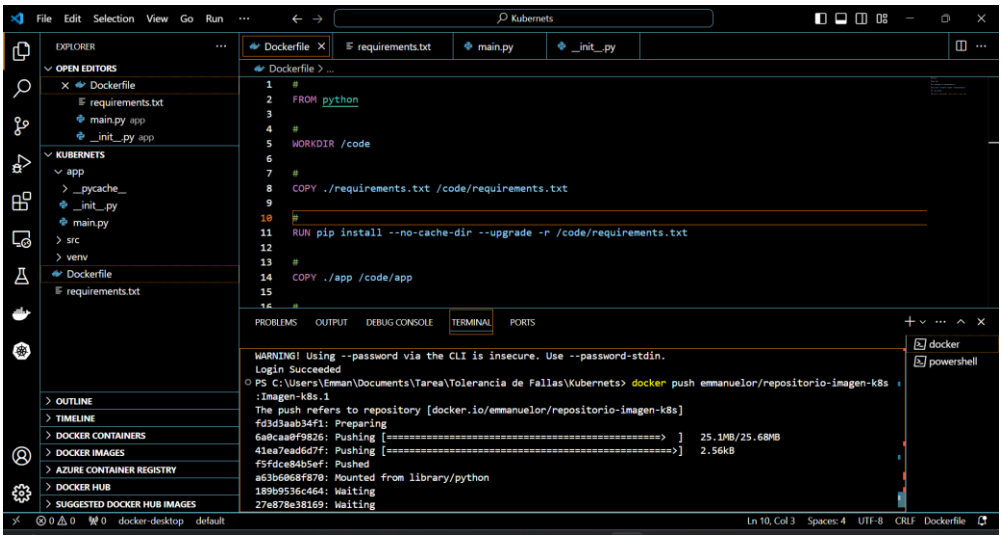
1. Imágenes creados y guardados localmente en Docker Desktop.

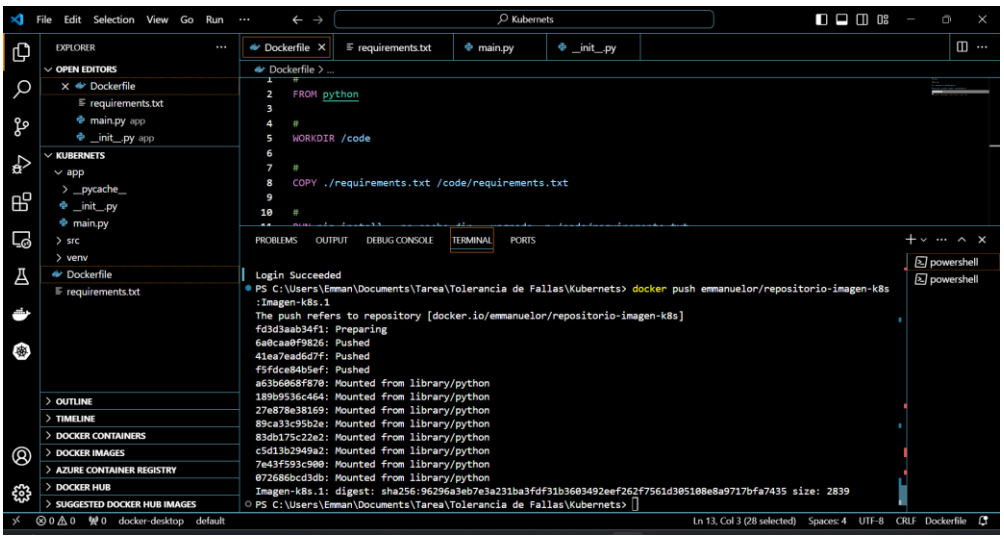


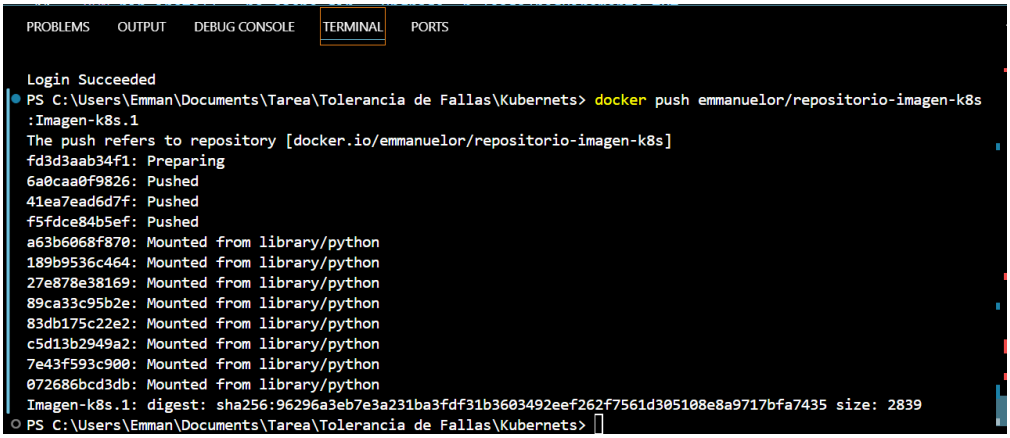
2. Hacer login a Docker Hub, y realizar el push de la imagen.

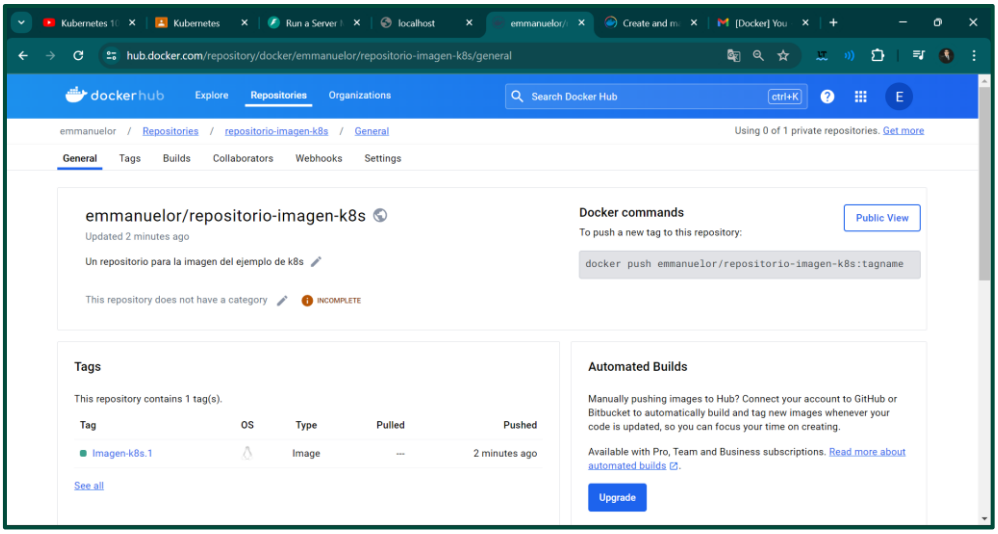





File Edit Selection View Go Run
Kubernets
EXPLORER
OPEN EDITORS
Dockerfile
requirements.txt
main.py app
__init__.py app
KUBERNETS
app
__pycache__
__init__.py
main.py
src
venv
Dockerfile
requirements.txt
OUTLINE
TIMELINE
DOCKER CONTAINERS
DOCKER IMAGES
AZURE CONTAINER REGISTRY
DOCKER HUB
SUGGESTED DOCKER HUB IMAGES
Dockerfile
requirements.txt
main.py
__init__.py
Dockerfile > ...
1 #
2 FROM python
3
4 #
5 WORKDIR /code
6
7 #
8 COPY ./requirements.txt /code/requirements.txt
9
10 #
11 RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt
12
13 #
14 COPY ./app /code/app
15
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
docker
powershell
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
PS C:\Users\Emman\Documents\Tarea\Tolerancia de Fallas\Kubernets> docker push emmanuelor/repositorio-imagen-k8s
:Imagen-k8s.1
The push refers to repository [docker.io/emmanuelor/repositorio-imagen-k8s]
fd3d3aab34f1: Preparing
6a0caa0f9826: Pushing [==========>] 25.1MB/25.68MB
41ea7ead6d7f: Pushing [==========>] 2.56kB
f5fdce84b5ef: Pushed
a63b6068f870: Mounted from library/python
189b9536c464: Waiting
27e878e38169: Waiting
docker-desktop default
Ln 10, Col 3 Spaces: 4 UTF-8 CRLF Dockerfile


File Edit Selection View Go Run
Kubernets
EXPLORER
OPEN EDITORS
Dockerfile
requirements.txt
main.py app
__init__.py app
KUBERNETS
app
__pycache__
__init__.py
main.py
src
venv
Dockerfile
requirements.txt
OUTLINE
TIMELINE
DOCKER CONTAINERS
DOCKER IMAGES
AZURE CONTAINER REGISTRY
DOCKER HUB
SUGGESTED DOCKER HUB IMAGES
Dockerfile
requirements.txt
main.py
__init__.py
Dockerfile > ...
2 FROM python
3
4 #
5 WORKDIR /code
6
7 #
8 COPY ./requirements.txt /code/requirements.txt
9
10 #
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
powershell
Login Succeeded
PS C:\Users\Emman\Documents\Tarea\Tolerancia de Fallas\Kubernets> docker push emmanuelor/repositorio-imagen-k8s
:Imagen-k8s.1
The push refers to repository [docker.io/emmanuelor/repositorio-imagen-k8s]
fd3d3aab34f1: Preparing
6a0caa0f9826: Pushed
41ea7ead6d7f: Pushed
f5fdce84b5ef: Pushed
a63b6068f870: Mounted from library/python
189b9536c464: Mounted from library/python
27e878e38169: Mounted from library/python
89ca33c95b2e: Mounted from library/python
83db175c22e2: Mounted from library/python
c5d13b2949a2: Mounted from library/python
7e43f593c900: Mounted from library/python
072686bcd3db: Mounted from library/python
Imagen-k8s.1: digest: sha256:96296a3eb7e3a231ba3fdf31b3603492eef262f7561d305108e8a9717bfa7435 size: 2839
PS C:\Users\Emman\Documents\Tarea\Tolerancia de Fallas\Kubernets>
docker-desktop default
Ln 13, Col 3 (28 selected) Spaces: 4 UTF-8 CRLF Dockerfile


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Login Succeeded
PS C:\Users\Emman\Documents\Tarea\Tolerancia de Fallas\Kubernets> docker push emmanuelor/repositorio-imagen-k8s
:Imagen-k8s.1
The push refers to repository [docker.io/emmanuelor/repositorio-imagen-k8s]
fd3d3aab34f1: Preparing
6a0caa0f9826: Pushed
41ea7ead6d7f: Pushed
f5fdce84b5ef: Pushed
a63b6068f870: Mounted from library/python
189b9536c464: Mounted from library/python
27e878e38169: Mounted from library/python
89ca33c95b2e: Mounted from library/python
83db175c22e2: Mounted from library/python
c5d13b2949a2: Mounted from library/python
7e43f593c900: Mounted from library/python
072686bcd3db: Mounted from library/python
Imagen-k8s.1: digest: sha256:96296a3eb7e3a231ba3fdf31b3603492eef262f7561d305108e8a9717bfa7435 size: 2839
PS C:\Users\Emman\Documents\Tarea\Tolerancia de Fallas\Kubernets>


hub.docker.com/repository/docker/emmanuelor/repositorio-imagen-k8s/general
docker hub
Explore
Repositories
Organizations
Search Docker Hub
ctrl+K
emmanuelor / Repositories / repositorio-imagen-k8s / General
Using 0 of 1 private repositories. Get more
General
Tags
Builds
Collaborators
Webhooks
Settings
emmanuelor/repositorio-imagen-k8s
Updated 2 minutes ago
Un repositorio para la imagen del ejemplo de k8s
This repository does not have a category
INCOMPLETE
Docker commands
To push a new tag to this repository:
Public View
docker push emmanuelor/repositorio-imagen-k8s:tagname
Tags
This repository contains 1 tag(s).
Tag
OS
Type
Pulled
Pushed
Imagen-k8s.1
Image
---
2 minutes ago
See all
Automated Builds
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.
Available with Pro, Team and Business subscriptions. Read more about automated builds.
Upgrade

emmanuelor/repositorio-imagen-k8s:Imagen-k8s.1

Delete Tag

MANIFEST DIGEST sha256:96296a3eb7e3a231ba3fdf31b3603492eef262f7561d305108e8a9717bfa7435

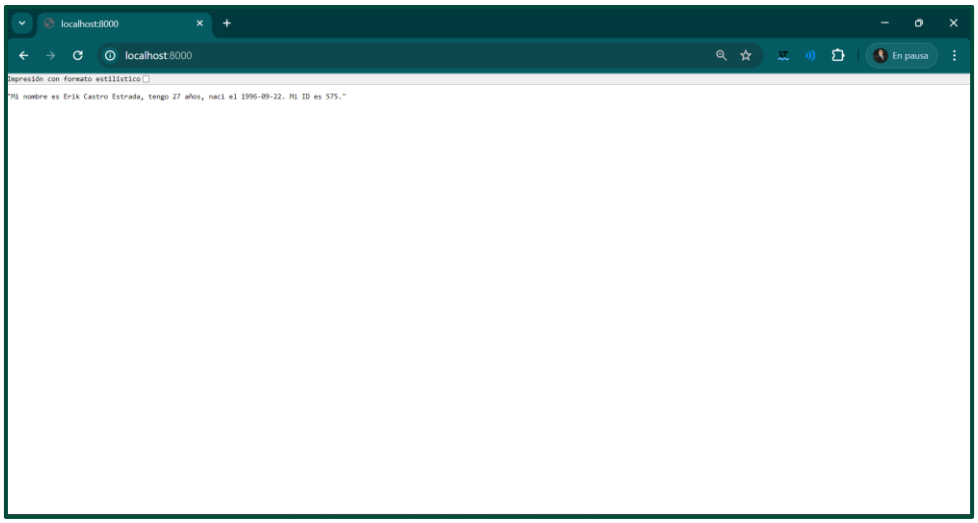
OS/ARCH	COMPRESSED SIZE	LAST PUSHED	TYPE	MANIFEST DIGEST
linux/amd64	372.12 MB	3 hours ago by emmanuelor	Image	sha256:96296a3e…

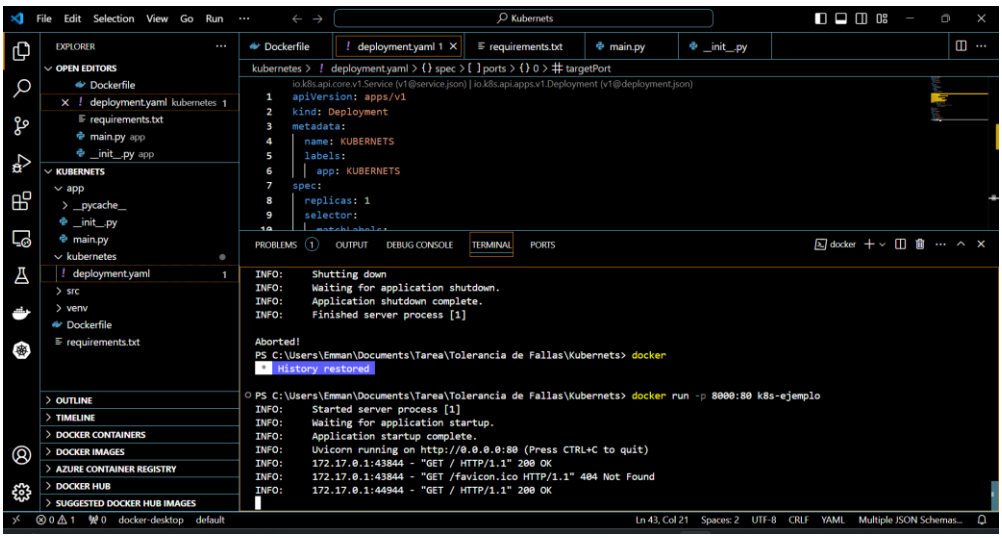
Image Layers | Vulnerabilities

IMAGE LAYERS

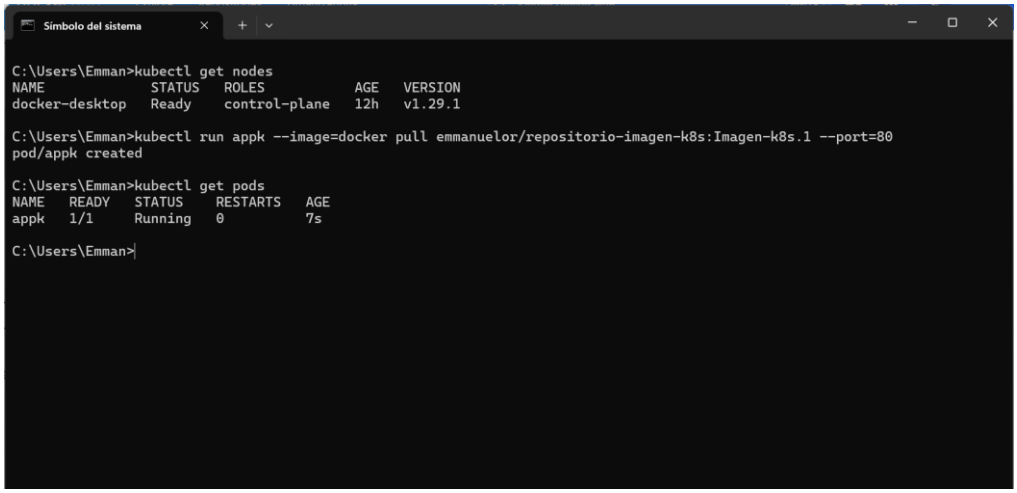
1	**ADD file ... in /**	**47.28 MB**
2	CMD ["bash"]	0 B
3	/bin/sh -c set -eux; apt-get	22.94 MB
4	/bin/sh -c set -eux; apt-get	61.17 MB
5	/bin/sh -c set -ex; apt-get	201.39 MB
6	ENV PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin	0 B
7	ENV LANG=C.UTF-8	0 B
8	RUN /bin/sh -c set -eux;	6.1 MB
9	ENV GPG_KEY=7169605F62C751356D054A26A821E680E5FA6305	0 B
10	ENV PYTHON_VERSION=3.12.3	0 B
11	RUN /bin/sh -c set -eux;	21.65 MB
12	RUN /bin/sh -c set -eux;	244 B
13	ENV PYTHON_PIP_VERSION=24.0	0 B
14	ENV PYTHON_GET_PIP_URL=https://github.com/pypa/get-pip/raw/dbf0c85f76fb6e1ab42aa672ffca6f0a675…	0 B
15	ENV PYTHON_GET_PIP_SHA256=dfe9fd5c28dc98b5ac17979a953ea550cec37ae1b47a5116007395bfacff2ab9	0 B
16	RUN /bin/sh -c set -eux;	2.64 MB
17	CMD ["python3"]	0 B
18	WORKDIR /code	94 B
19	COPY ./requirements.txt /code/requirements.txt # buildkit	182 B
20	RUN /bin/sh -c pip install	8.95 MB
21	COPY ./app /code/app # buildkit	3.65 KB
22	CMD ["uvicorn" "app.main:app" "--host" "0.0.0.0"	0 B

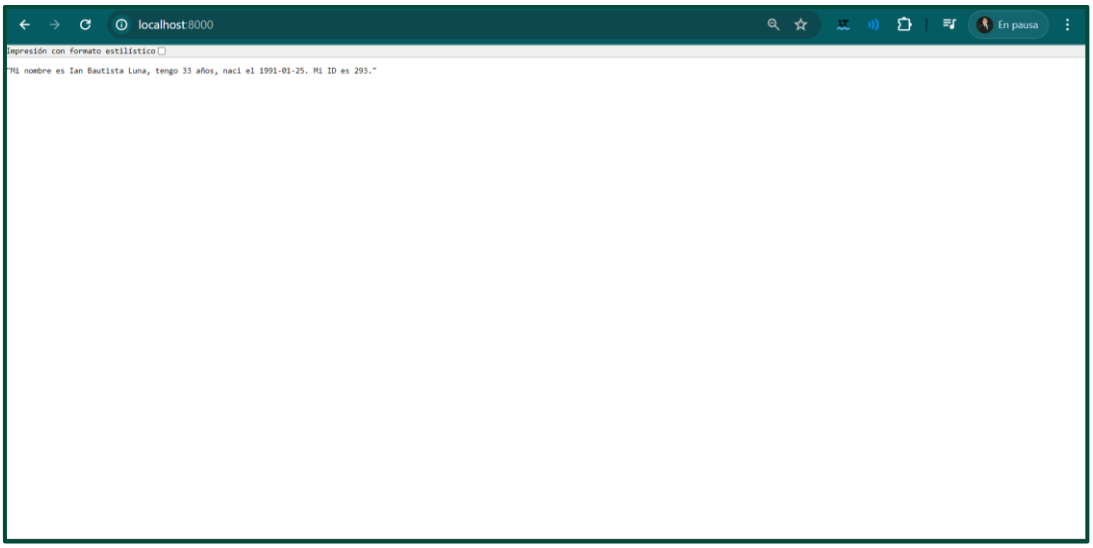
3. Ejecución de la imagen en Docker usando Docker Desktop como servidor.

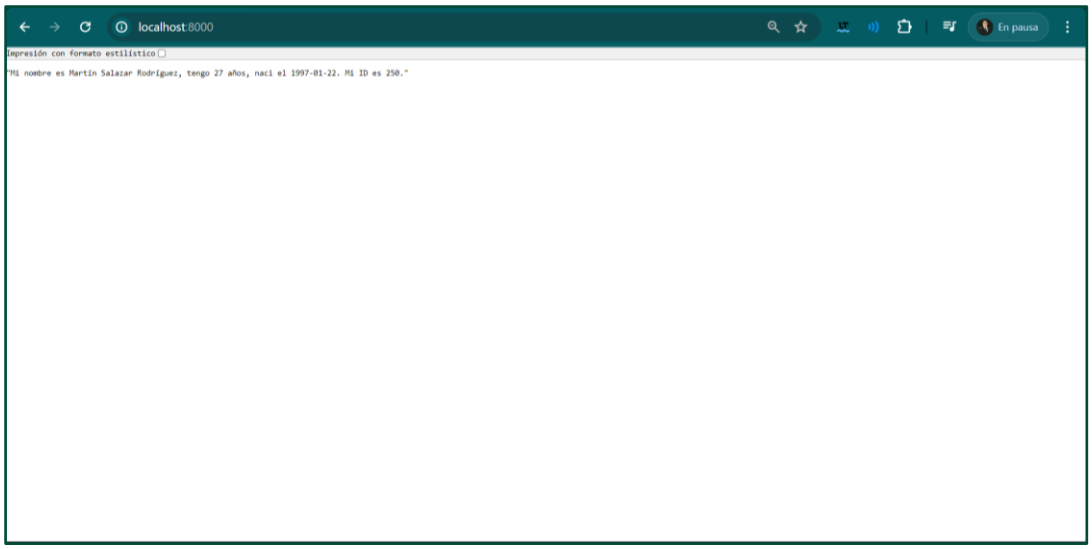


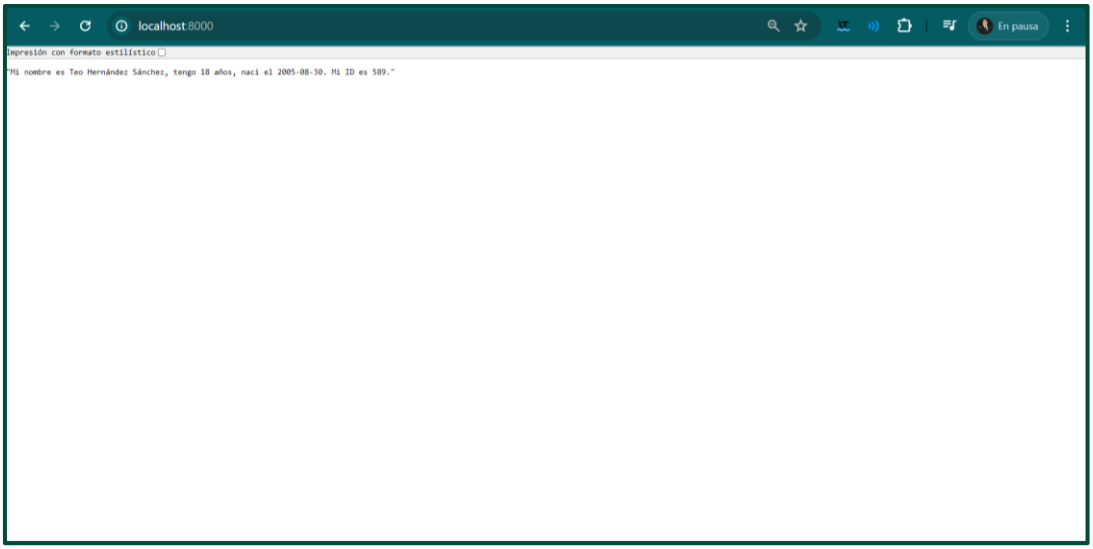


3. Creación y ejecución del pod, y ejecución con el cluster de minikube.








localhost:8000
En pausa
Impresión con formato estilístico
"Mi nombre es Teo Hernández Sánchez, tengo 18 años, nací el 2005-08-30. Mi ID es 589."