



Docker

Computación Tolerante a Fallas

Alumno:

Código:

Profesor(a): Dr. Michel Emanuel López Franco

Sección: D06

Fecha de Entrega: 24 de octubre de 2022

Docker

Es una plataforma de software que empaqueta software en unidades estandarizadas llamadas contenedores las cuales contienen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. De esta forma, es posible ejecutar aplicaciones rápidamente, además de implementar y ajustar la escala de estas rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará.

Cada contenedor contiene una serie de instrucciones que indicaran la forma en que debe arrancar.

Ejemplo

Para esta actividad, decidí implementar un programa en Python que muestra, en forma de un mensaje de alerta, un dato aleatorio sobre Chuck Norris obtenido desde una API. Hice uso de un Flow de prefect para facilitar la repetición del programa.

Dentro de mi IDE, cree un archivo de Python donde importo todos los módulos que voy a necesitar para trabajar.

```
import requests
from tkinter import *
from tkinter import messagebox

from prefect import task, Flow, Parameter
from prefect.schedules import IntervalSchedule
from datetime import timedelta
```

Luego, inicializamos las variables y datos de cabecera necesarios para hacer las consultas a la AP, la raíz del modulo Tkinter para poder mostrar los mensajes en la pantalla, y una agenda de intervalos para que los mensajes se muestren cada minuto.

```
url = "https://matchilling-chuck-norris-jokes-v1.p.rapidapi.com/jokes/random"

root = Tk ()

headers = {
    "accept": "application/json",
    "X-RapidAPI-Key": "7a692470demshde86b2352572440p1a7720jsn888da3681c6c",
    "X-RapidAPI-Host": "matchilling-chuck-norris-jokes-v1.p.rapidapi.com"
}

scheduleData = IntervalSchedule (interval=timedelta (minutes=1))
```

Ahora vienen las funciones del Flow ETL, donde se extraen los datos de la API, se formatean del JSON a una cadena de texto, y se muestran en un mensaje usando la clase *messagebox* de Tkinter. Se establece que, si ocurre un error, lo intente de nuevo un máximo de 10 veces con 15 segundos de retraso entre cada intento.

```
# Extraer
@task(max_retries=10, retry_delay=timedelta (seconds=15))
def obtenerDatos ():
    response = requests.request("GET", url, headers=headers)

    formato = response.json ()

    return formato

# Transformar
@task
def transformarDatos (datos):
    return datos["value"]

# Almacenar
@task
def almacenarDatos (datoChuck):
    messagebox.showinfo(message=datoChuck, title="Dato")
```

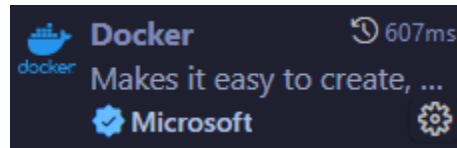
Finalmente, creamos el Flow y lo ejecutamos.

```
def crearFlow (schedule):
    # Agregamos la agenda como parametro a Flow.
    with Flow ("ETL Flow", schedule) as flow:
        datos = obtenerDatos ()
        contenido = transformarDatos (datos)
        almacenarDatos (contenido)

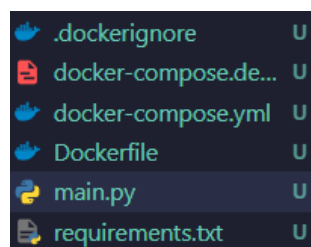
    return flow

flow = crearFlow (scheduleData)
# flow.visualize ()
flow.run ()
```

Lo siguiente es crear una aplicación dentro de Docker para guardar nuestro programa. Visual Studio Code ofrece una extensión de Docker muy potente que facilita bastante crear los archivos pertinentes y las imágenes.



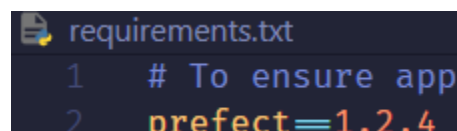
Con un simple comando, podemos agregar todos los archivos necesarios para crear la imagen a nuestro espacio de trabajo.



Incluso rellena el Dockerfile con la información necesaria.

```
Dockerfile > ...
1 # For more information, please refer to https://aka.ms/vscode-docker-python
2 FROM python:3.8-slim
3
4 # Keeps Python from generating .pyc files in the container
5 ENV PYTHONDONTWRITEBYTECODE=1
6
7 # Turns off buffering for easier container logging
8 ENV PYTHONUNBUFFERED=1
9
10 # Install pip requirements
11 COPY requirements.txt .
12 RUN python -m pip install -r requirements.txt
13
14 WORKDIR /app
15 COPY . /app
16
17 # Creates a non-root user with an explicit UID and adds permission to access the /app folder
18 # For more info, please refer to https://aka.ms/vscode-docker-python-configure-containers
19 RUN adduser --u 5678 --disabled-password --gecos "" appuser 66 chown -R appuser /app
20 USER appuser
21
22 # During debugging, this entry point will be overridden. For more information, please refer to https://aka.ms/vscode-docker-python-deb
23 CMD ["python", "main.py"]
```

Dentro del archivo *requirements.txt* es necesario incluir el nombre de aquellos módulos no nativos de Python y que son necesarios para que funcione el programa. Para este caso, solo es *prefect*.



Usamos el comando *build image* para construir la imagen.

```
>docker bui
```

Docker Images: Build Image...

```
> docker build --rm --pull -f "C:
```

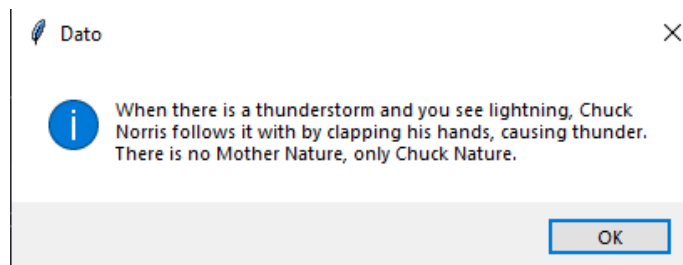
```
#12 exporting to image
#12 sha256:e8c613e07b0b7ff33893b694f7759a10d42e180f2b4dc349fb57dc6b71dcab00
#12 exporting layers
#12 exporting layers 0.9s done
#12 writing image sha256:2720eb6bf15b65e5767824ddf801e42fc9b2e208b0b87192015ea4b7ca25a56a done
#12 naming to docker.io/library/docker:latest done
#12 DONE 0.9s
```

Dentro del GUI de Docker, la imagen se ve de la siguiente forma:





NAME ↑	TAG	IMAGE ID	CREATED	SIZE
docker	latest	2720eb6bf15b	8 minutes ago	191.28 MB

Con el comando *docker run docker* podemos ejecutar nuestra aplicación.

NAME ↑		TAG	IMAGE ID	CREATED	SIZE
docker	IN USE	latest	74daab59bf0f	1 minute ago	191.28 MB



Automáticamente se crea un contenedor.

<input type="checkbox"/>	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input type="checkbox"/>	 admiring_stonebraker 09c8960e02eb	docker:latest	Running	-	2 minutes agc	  

Conclusiones

Esta actividad fue bastante divertida. Gracias a que en la práctica pasada había tenido un acercamiento a Docker, me resulto sencillo implementarlo con otro programa. Además, la extensión que ofrece Visual Studio Code es increíblemente útil, ya que te ahorra el escribir muchas líneas de código y aprenderte comandos de la terminal, los cuales pueden resultar complejos.

Docker es una herramienta increíblemente práctica, y espero poder utilizarla en mi vida profesional y en futuros proyectos.

Bibliografía

Contenedores de Docker | ¿Qué es Docker? | AWS. (s. f.). Amazon Web Services, Inc. Recuperado 10 de octubre de 2022, de <https://aws.amazon.com/es/docker/>