

Universidad Tecnológica del Norte de
Guanajuato

Área: Tecnologías de la Información

Programa educativo: Infraestructura de
redes digitales

Asignatura: Programación de Redes

UNIDAD II

Profesor: Gabriel Barrón Rodríguez

PRACTICA
Build your python images

Alumno: Canchola Ramírez Mariana

Dolores Hidalgo C.I.N 08 noviembre 2022

Instalación de Docker Desktop

Build your Python image

Crear el archivo daemon.json

```
sudo touch /etc/docker/daemon.json
```

```
marianarmz@marianarmz:~/test-docker$ sudo touch /etc/docker/daemon.json
```

```
sudo chmod 777 daemon.json
```

```
marianarmz@marianarmz:~/test-docker$ sudo chmod 777 /etc/docker/daemon.json
```

```
systemctl --user start docker-desktop
```

```
marianarmz@marianarmz:~/test-docker$ systemctl --user start docker-desktop
```

Crea un directorio llamado python-docker en el lugar que tu indiques

```
marianarmz@marianarmz:~$ mkdir test-docker
```

Cambiar al directorio recién creado

```
cd test-docker
```

```
marianarmz@marianarmz:~$ cd test-docker/
```

Instalar los módulos que usaremos para Python

```
pip3 install Flask
```

```
marianarmz@marianarmz:~/test-docker$ pip3 install Flask
```

```
pip3 freeze | grep Flask >> requirements.txt
```

```
marianarmz@marianarmz:~/test-docker$ pip freeze | grep Flask >> requirements.txt
```

```
touch app.py
```

```
touch app.py
```

Abrir Visual Studio Code

Agregar el siguiente código al archivo app.py

```
app.py  X
app.py > ...
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  @app.route('/')
6  def hello_world():
7      return "Hello Mariana, Docker!"
8
```

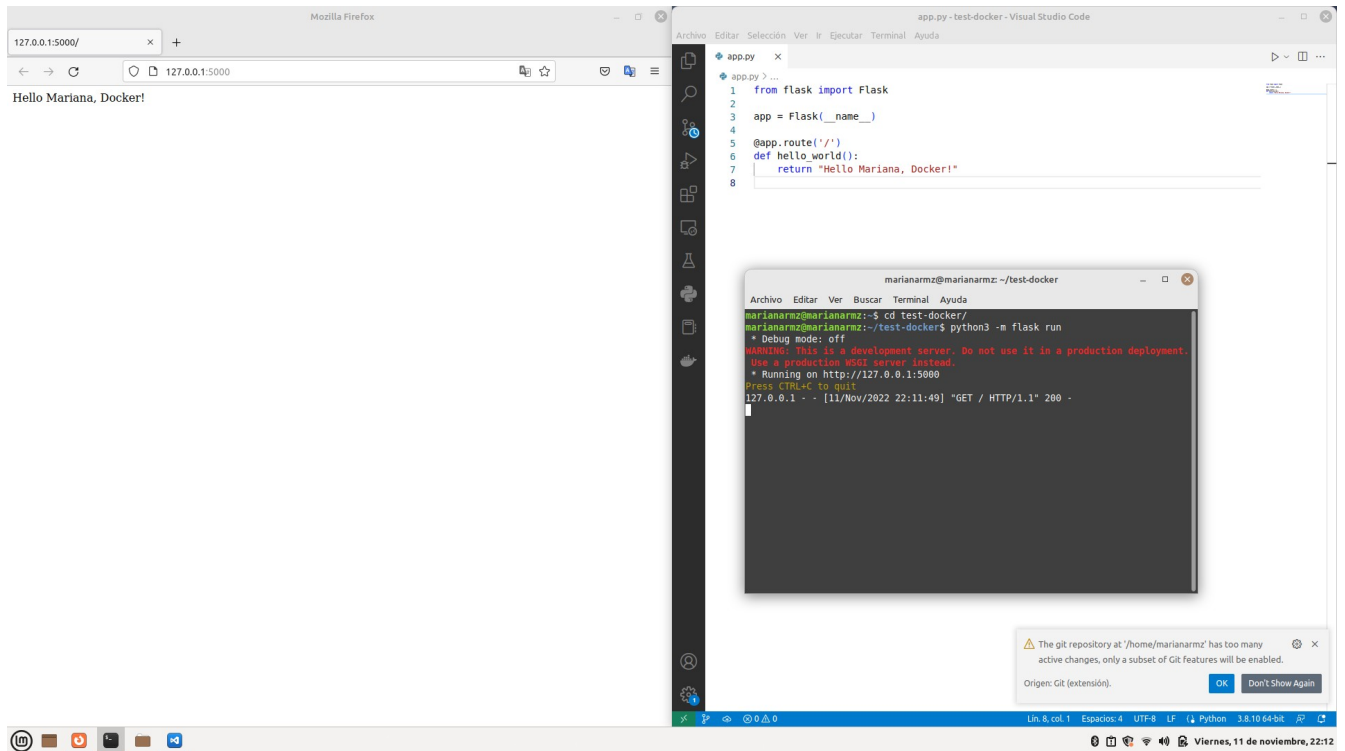
PROBLEMAS CONSOLA DE DEPURACIÓN SALIDA TERMINAL JUPYTER

```
/bin/python3 /home/marianarmz/test-docker/app.py
marianarmz@marianarmz:~/test-docker$ /bin/python3 /home/marianarmz/test-docker/app.py
```

Prueba la aplicación ejecutando el comando en la terminal

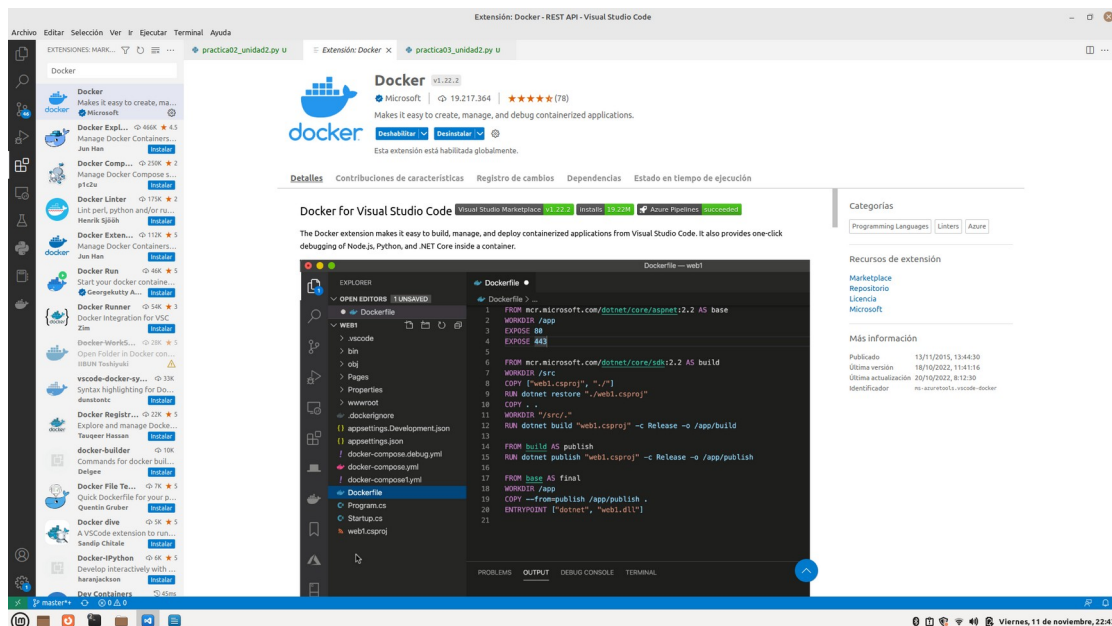
```
marianarmz@marianarmz:~/test-docker$ python3 -m flask run
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [11/Nov/2022 22:11:49] "GET / HTTP/1.1" 200 -
```

Abre un navegador web y verificar funcionamiento



Create a Dockerfile for Python

Primeramente instalar extensión para Docker en Visual Studio Code



Creamos el Dockerfile e introducimos la información que nos ayudará a ejecutar nuestras imágenes.

```
test-docker > Dockerfile > ...
1  # syntax=docker/dockerfile:1
2
3  FROM python:3.8-slim-buster
4
5  WORKDIR /app
6
7  COPY requirements.txt requirements.txt
8  RUN pip3 install -r requirements.txt
9
10 COPY . .
11
12 CMD [ "python3", "-m", "flask", "run", "--host=0.0.0.0" ]
13
```

Construir la imagen:

```
root@marianarmz:/home/marianarmz/test-docker# docker build --tag python-docker .
```

Esperamos a la creación de la imagen

```
root@marianarmz:/home/marianarmz/test-docker# docker build --tag python-docker .
Sending build context to Docker daemon  10.24kB
Step 1/6 : FROM python:3.8-slim-buster
3.8-slim-buster: Pulling from library/python
4508a762c546: Pull complete
6aa89787764f: Pull complete
9377b4a8d06d: Pull complete
d0e5195f1f58: Pull complete
f8d7123acbc: Pull complete
Digest: sha256:2faab80de0ebd11bb549be5b7b626ad23fcd8fe7998ad02a708381f1808a3fd5
Status: Downloaded newer image for python:3.8-slim-buster
--> d55c26ea3903
Step 2/6 : WORKDIR /app
--> Running in 49d92f2e99a4
Removing intermediate container 49d92f2e99a4
--> 32a926036540
Step 3/6 : COPY requirements.txt requirements.txt
--> bc40f06019e
Step 4/6 : RUN pip3 install -r requirements.txt
--> Running in 4e9166327ae7
Collecting Flask==2.2.2
  Downloading Flask-2.2.2-py3-none-any.whl (101 kB)
    101.5/101.5 KB 1.3 MB/s eta 0:00:00
Collecting itsdangerous==2.0
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Werkzeug==2.2.2
  Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
    232.7/232.7 KB 3.1 MB/s eta 0:00:00
Collecting Jinja2==3.0
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
    133.1/133.1 KB 3.6 MB/s eta 0:00:00
Collecting click==8.0
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
    96.6/96.6 KB 3.7 MB/s eta 0:00:00
Collecting importlib-metadata==3.6.0
  Downloading importlib_metadata-5.0.0-py3-none-any.whl (21 kB)
Collecting zipp==0.5
  Downloading zipp-3.10.0-py3-none-any.whl (6.2 kB)
Collecting MarkupSafe==2.0
  Downloading MarkupSafe-2.1.1-cp38-cp38-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (25 kB)
Installing collected packages: zipp, MarkupSafe, itsdangerous, click, Werkzeug, Jinja2, importlib-metadata, Flask
Successfully installed Flask-2.2.2 Jinja2-3.1.2 MarkupSafe-2.1.1 Werkzeug-2.2.2 click-8.1.3 importlib-metadata-5.0.0 itsdangerous-2.1.2 zipp-3.10.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
NOTICE: You are using pip version 22.0.4; however, version 22.3.1 is available.
You should consider upgrading via the '/usr/local/bin/python -m pip install --upgrade pip' command.
Removing intermediate container 4e9166327ae7
--> 1c7940779204
Step 5/6 : COPY . .
--> c28f10d4baf8
Step 6/6 : CMD [ "python3", "-m", "flask", "run", "--host=0.0.0.0" ]
--> Running in ec6e20ba7323
Removing intermediate container ec6e20ba7323
--> a2d78305741a
Successfully built a2d78305741a
Successfully tagged python-docker:latest

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
root@marianarmz:/home/marianarmz/test-docker# docker image
```

Visualizamos las imágenes

```
root@marianarmz:/home/marianarmz/test-docker# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
python-docker	latest	a2d70305741a	23 seconds ago	128MB
python	3.8-slim-buster	d55c26ea3903	2 weeks ago	117MB
hello-world	latest	feb5d9fea6a5	13 months ago	13.3kB

Creamos un tag para la imagen

```
root@marianarmz:/home/marianarmz/test-docker# docker tag python-docker:latest python-docker:v1.0.0
```

Y volvemos a revisar nuestras imágenes

```
root@marianarmz:/home/marianarmz/test-docker# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
python-docker	latest	a2d70305741a	41 seconds ago	128MB
python-docker	v1.0.0	a2d70305741a	41 seconds ago	128MB
python	3.8-slim-buster	d55c26ea3903	2 weeks ago	117MB
hello-world	latest	feb5d9fea6a5	13 months ago	13.3kB

Removemos el tag que acabamos de crear

```
root@marianarmz:/home/marianarmz/test-docker# docker rmi python-docker:v1.0.0
Untagged: python-docker:v1.0.0
root@marianarmz:/home/marianarmz/test-docker# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
python-docker	latest	a2d70305741a	About a minute ago	128MB
python	3.8-slim-buster	d55c26ea3903	2 weeks ago	117MB
hello-world	latest	feb5d9fea6a5	13 months ago	13.3kB

Por ultimo ejecutamos la imagen en el contenedor

```
root@marianarmz:/home/marianarmz/test-docker# docker run python-docker
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
```

Abrimos una nueva terminal y ejecutamos el comando curl localhost:5000.

Como puede ver, nuestro curl El comando falló porque se rechazó la conexión a nuestro servidor.

Por lo cual asignaremos el puerto 8000

```
root@marianarmz:/home/marianarmz/test-docker# docker run --publish 8000:5000 python-docker
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.17.0.1 - - [12/Nov/2022 04:40:24] "GET / HTTP/1.1" 200 -
```

Y haremos lo mismo, abriremos una terminal y ejecutaremos el comando curl localhost:8000

```
marianarmz@marianarmz:~/test-docker$ curl localhost:8000  
Hello Mariana, Docker!marianarmz@marianarmz:~/test-docker$
```

Preguntas a responder

-¿Cómo defines Docker?

Docker es una aplicación muy fácil y sencilla, al principio tuve problemas para crear mis imágenes pero después pude ejecutarlo sin dificultad.

¿Cuáles son los beneficios de utilizar un contenedor Docker?

Sus beneficios son la creación de contenedores fácil y rápido, tiene un funcionamiento constante y mucha eficiencia al momento de trabajar en el.

¿Cuáles son las características clave de Docker?

Tiene fiabilidad.

Tiene la capacidad de desplegar múltiples contenedores en un mismo equipo.

Es fácil de usar.

Es posible encapsular todo el entorno de trabajo.

¿Cuáles son las principales desventajas de Docker?

Se requiere tener mínimo la versión de Kernel 3.8.

Algunas de sus versiones dan error ya que se encuentran en constante desarrollo.

Para el sistema operativo Windows aún se encuentra en desarrollo.

Solo soporta sistemas operativos linux de 64 bits.

¿Qué es una imagen de Docker?

Una imagen de Docker es una plantilla que solo se utiliza como lectura para definir su contenedor.

Esta contiene el código que se ejecutará, incluidas así las definiciones de bibliotecas o dependencias que necesita el código.

¿Cuáles son las instrucciones habituales en Dockerfile?

Son:

ENV

Variable de entorno para el proceso de construcción y establecer vida del contenedor.

WORKDIR

Cambiar de directorio actual.

FROM

Establecer imagen base.

USER

Cambiar usuario y pertenencia al grupo.

RUN

Ejecutar el comando de Image durante el proceso de construcción.

CMD

Establecer argumentos estándar para el inicio del contenedor.

VOLUME

Integrar como volumen directorio de Image al iniciar el contenedor en el sistema anfitrión.

¿Cuáles son los estados en un contenedor Docker?

Created

Restarting

Removing

Running

Unhealthy

Exited

Paused

Dead