



Juan Pablo Palma Apoderado

NOMBRE

TSU Infraestructura de Redes Digitales

CARRERA

GIR0441

GRUPO

Programación de Redes

MATERIA

Build your python images

PRACTICA

**Gabriel Barrón Rodríguez**

DOCENTE

Universidad Tecnológica del Norte de Guanajuato

LUGAR

8 de noviembre del 2022

FECHA:

# Instalación de Docker Desktop Automatización de Infraestructura Digital I

## Build your Python image

Para una mayor referencia véase enlace  
Crear el archivo daemon.json

sudo touch /etc/docker/daemon.json

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

sudo chmod 777 daemon.json

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

systemctl --user start docker-desktop

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

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

```
jpalma@PAJP:~$ mkdir test-docker
```

cambiar al directorio recién creado

cd test-docker

```
jpalma@PAJP:~$ cd test-docker/
```

Instalar los módulos que usaremos para Python

pip3 install Flask

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

```
jpalma@PAJP:~/test-docker$ pip3 install Flask
Defaulting to user installation because normal site-packages is not writeable
Collecting Flask
  Downloading Flask-2.2.2-py3-none-any.whl (101 kB)
    101.5/101.5 kB 302.5 kB/s eta 0:00:00
Requirement already satisfied: click>=8.0 in /usr/lib/python3/dist-packages (from Flask) (8.0.3)
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 379.5 kB/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 994.7 kB/s eta 0:00:00
Requirement already satisfied: MarkupSafe>=2.0 in /usr/lib/python3/dist-packages (from Jinja2>=3.0->Flask) (2.0.1)
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.1-cp310-cp310-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (25 kB)
Installing collected packages: MarkupSafe, itsdangerous, Werkzeug, Jinja2, Flask
Successfully installed Flask-2.2.2 Jinja2-3.1.2 MarkupSafe-2.1.1 Werkzeug-2.2.2 itsdangerous-2.1.2
```

pip3 freeze | grep Flask >> requirements.txt

```
jpalma@PAJP:~/test-docker$ pip3 freeze | grep Flask >> requirements.txt
```

touch app.py

```
jpalma@PAJP:~/test-docker$ touch app.py
```

Abrir Visual Studio Code

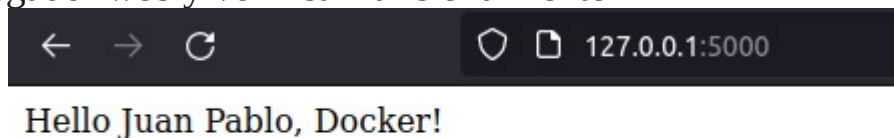
Agregar el siguiente código al archivo app.py

```
1
2  from flask import Flask
3
4  app = Flask(__name__)
5
6  @app.route('/')
7
8  def hello_world():
9      return 'Hello Juan Pablo, Docker!'
```

Prueba la aplicación ejecutando el comando en la terminal

```
jpalma@PAJP:~/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 in
stead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [03/Nov/2022 14:41:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [03/Nov/2022 14:41:18] "GET /favicon.ico HTTP/1.1" 404 -
```

Abre un navegador web y verificar funcionamiento

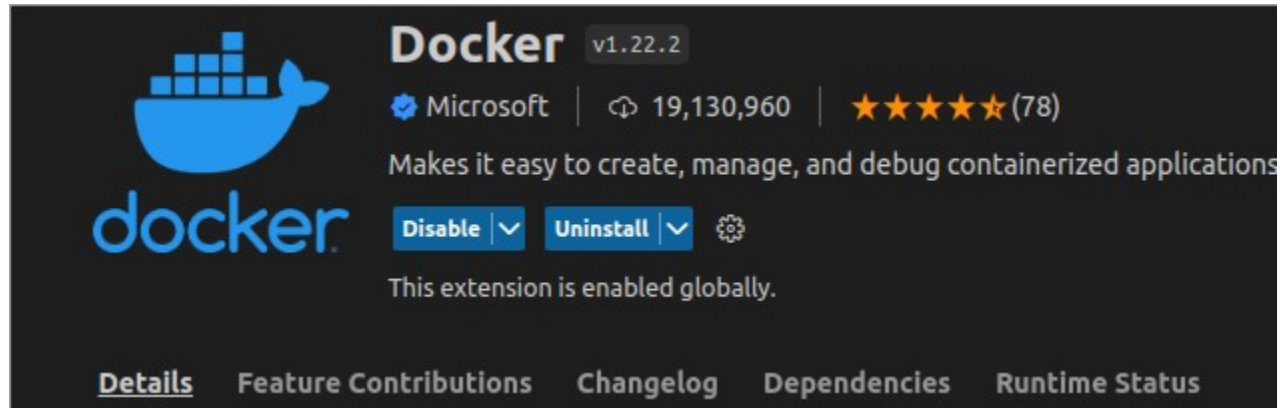


← → ↻ 🔒 127.0.0.1:5000

Hello Juan Pablo, Docker!

## Create a Dockerfile for Python

Instalar la extensión para Docker dentro de Visual Studio Code



Construye la imagen para alojar la aplicación

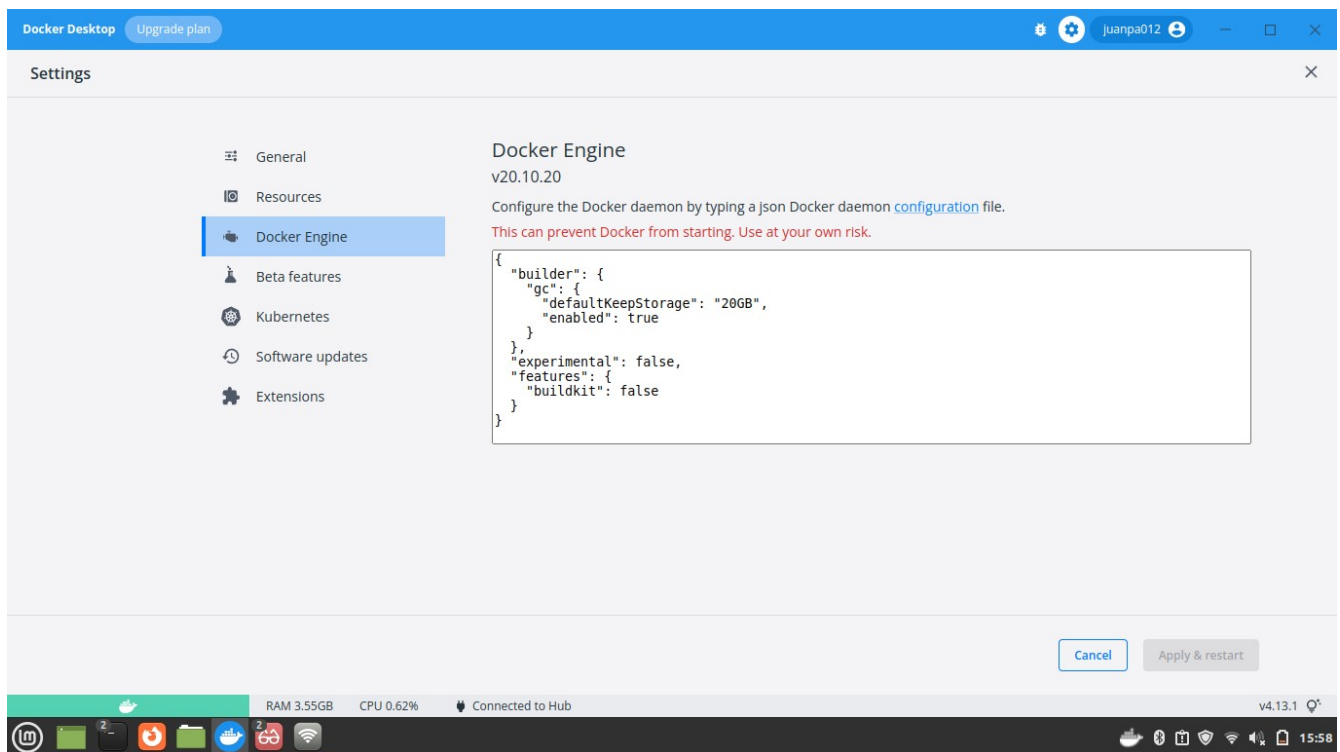
```
docker build --tag python-docker .
```

Esperar a la creación de la imagen

```
(.venv) jpalma@PAJP:~/test-docker$ docker build --tag python-docker .
Sending build context to Docker daemon 21.12MB
Step 1/6 : FROM python:3.8-slim-buster
3.8-slim-buster: Pulling from library/python
4500a762c546: Pull complete
eaa89787764f: Pull complete
9377b4a0d06d: Pull complete
d0e5195f1f50: Pull complete
f0d77123acbc: Pull complete
Digest: sha256:2faab08dbeb0d11bb549be5b7b626ad23fcd0fe7998ad02a708381f1800a3fd5
Status: Downloaded newer image for python:3.8-slim-buster
--> d55c26ea3903
Step 2/6 : WORKDIR /app
--> Running in 7630097220ca
Removing intermediate container 7630097220ca
--> 3f202dd0f60e
Step 3/6 : COPY requirements.txt requirements.txt
--> b990af638776
Step 4/6 : RUN pip3 install -r requirements.txt
--> Running in 4367b049412b
Collecting click==8.1.3
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
    _____ 96.6/96.6 KB 387.7 kB/s eta 0:00:00
Collecting Flask==2.2.2
  Downloading Flask-2.2.2-py3-none-any.whl (101 kB)
    _____ 101.5/101.5 KB 609.3 kB/s eta 0:00:00
Collecting itsdangerous==2.1.2
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Jinja2==3.1.2
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
    _____ 133.1/133.1 KB 508.5 kB/s eta 0:00:00
Collecting MarkupSafe==2.1.1
  Downloading MarkupSafe-2.1.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Collecting Werkzeug==2.2.2
  Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
    _____ 232.7/232.7 KB 961.8 kB/s eta 0:00:00
Collecting importlib-metadata>=3.6.0
  Downloading importlib-metadata-5.0.0-py3-none-any.whl (21 kB)
```

Nota : En mi caso yo tuve errores al momento de realizar la imagen pero lo solucione Cambiando

"buildkit": true A "buildkit": false



Ver las imagenes locales mediante el siguiente comando

```
(.venv) jpalma@PAJP:~/test-docker$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
python-docker	latest	39f803b8b349	41 seconds ago	148MB
python	3.8-slim-buster	d55c26ea3903	2 weeks ago	117MB

Crear un tag para la imagen

```
(.venv) jpalma@PAJP:~/test-docker$ docker tag python-docker:latest python-docker:v1.0.0
```

```
(.venv) jpalma@PAJP:~/test-docker$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
python-docker	latest	39f803b8b349	15 seconds ago	148MB
python-docker	v1.0.0	39f803b8b349	15 seconds ago	148MB
python	3.8-slim-buster	d55c26ea3903	2 weeks ago	117MB

Removamos el TAG

```
(.venv) jpalma@PAJP:~/test-docker$ docker rmi python-docker:v1.0.0
```

Untagged: python-docker:v1.0.0



## Ejecuta la imagen en el contenedor

```
(.venv) jpalma@PAJP:~/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
```

Abrir un navegador web y teclea la URL `http://localhost:5000/`

Abrir una terminal y teclear

`curl localhost:5000`

```
(.venv) jpalma@PAJP:~/test-docker$ curl localhost:5000
curl: (7) Failed to connect to localhost port 5000 after 0 ms: Connection refused
```

Teclear nuevamente el comando

`docker run --publish 8000:5000 python-docker`

```
(.venv) jpalma@PAJP:~/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 - - [08/Nov/2022 21:56:24] "GET / HTTP/1.1" 200 -
```

Prueba

```
jpalma@PAJP:~/test-docker$ curl localhost:8000
Hello Juan Pablo, Docker!jpalma@PAJP:~/test-docker$
```

Listar contenedores con el siguiente comando

`docker ps`

```
(.venv) jpalma@PAJP:~/test-docker$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
64ebacf1c49   python-docker:latest   "python3 -m flask ru..." 14 seconds ago   Up 11 seconds               bold_mclaren
6a6cf326063e   python-docker          "python3 -m flask ru..." 6 hours ago     Up 5 seconds               0.0.0.0:8000->5000/tcp   determined_lumiere
```

Parar el contenedor a través del ID o través de nombre del contenedor

`docker stop`

```
(.venv) jpalma@PAJP:~/test-docker$ docker stop 64ebacf1c49
64ebacf1c49
(.venv) jpalma@PAJP:~/test-docker$ docker stop 6a6cf326063e
6a6cf326063e
(.venv) jpalma@PAJP:~/test-docker$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
(.venv) jpalma@PAJP:~/test-docker$
```

Preguntas a responder

❖ ¿Cómo defines Docker?

Docker es muy sencillo y simple para la creación de contenedores.

❖ ¿Cuáles son los beneficios de utilizar un contenedor Docker?

La creación de contenedores con un menor comandos y su uso en servidores.

❖ ¿Cuáles son las características clave de Docker?

-No tener que preocuparse por la los comandos si no que trabajara.

-Para realizar testing

-Muy sencillo de crear y eliminar contenedores.

-No son maquinas virtuales .

-Se puede crear con otras aplicaciones.

-El tiempo que tarda en ejecutarse los contenedores es menor.

❖ ¿Cuáles son las principales desventajas de Docker?

-Se requiere un kernel 3.8

-Lagunas versiones de coker generan un error debido a que se encuentra en desarrollo.

-Solo soporta sistemas operativos de 64 Bits en Linux.

-Para el sistema operativo de Windows se encuentra en desarrollo

❖ ¿Qué es una imagen de Docker?

Una imagen de Docker es un archivo o archivo de varias capas que se utiliza para ejecutar código en un contenedor de Docker.

❖ ¿Cuáles son las instrucciones habituales en Dockerfile?

FROM — Especifica la base para la imagen.

ENV — Establece una variable de entorno persistente.

ARG — Permite definir una variable usable en el resto del Dockerfile con la sintaxis \${NOMBRE\_DEL\_ARG}

RUN — Ejecuta el comando especificado. ...

COPY — Copia archivos y directorios al contenedor.

❖ ¿Cuáles son los estados en un contenedor Docker?

Created: Esto significa que el contenedor se esta creando. ...

Restarting: El contenedor se está reiniciando.

Running: El contenedor está funcionando.

Removing: El contenedor se esta elimiando.



Paused: El contenedor se encuentra en estado pausado.