Taller de Docker

Mi primer Dockerfile

Creamos nuestro build context:

```
Diplomado on ?docker
[→ mkdir -p hello-world

Diplomado on ?docker
[→ cd hello-world

Diplomado/hello-world on ?docker
[→ echo "hello" > hello
```

Dentro de este directorio crearemos un archivo llamado Dockerfile con este contenido:

```
EXPLORER                    ...        Dockerfile ×
OPEN EDITORS                           Dockerfile > ...
  × Dockerfile                          1  FROM busybox
HELLO-WORLD                             2  COPY /hello /
  Dockerfile                            3  RUN cat /hello
```

Ahora para crear nuestra imagen usaremos docker build:

```
docker build -t helloapp:v1 .
```

```
Diplomado/hello-world on ?docker [?] on v26.0.0
→ docker build --no-cache --progress plain -t helloapp:v1 .
#0 building with "desktop-linux" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 77B done
#1 DONE 0.0s

#2 [internal] load metadata for docker.io/library/busybox:latest
#2 DONE 0.4s

#3 [internal] load .dockerignore
#3 transferring context: 2B done
#3 DONE 0.0s

#4 [1/3] FROM docker.io/library/busybox:latest@sha256:9ae97d36d26566ff84e8893c64a6dc4fe8ca6d1144bf5b87b2b85a32def253c7
#4 CACHED

#5 [internal] load build context
#5 transferring context: 26B done
#5 DONE 0.0s

#6 [2/3] COPY hello /
#6 DONE 0.0s

#7 [3/3] RUN cat /hello
#7 0.125 hello
#7 DONE 0.1s

#8 exporting to image
#8 exporting layers 0.0s done
#8 writing image sha256:ce64613a8b899bbc41fe65aa9514f084d546c66c0fe1aa5627bb6b04309ed1d3 done
#8 naming to docker.io/library/helloapp:v1 done
#8 DONE 0.1s
```

Y podremos ver que una nueva imagen está instalada en nuestro equipo:

```
Diplomado/hello-world on ?docker [?] on v26.0.0 took 3s
→ docker images
REPOSITORY        TAG       IMAGE ID       CREATED          SIZE
helloapp          v1        59f01fd393bc   40 seconds ago   4.26MB
```

´

Creando aplicaciones en contenedores

Comenzamos creando un nuevo build context:

```
Diplomado on ?docker [?]
→ mkdir friendlyhello

Diplomado on ?docker [?]
→ cd friendlyhello
```

El código de la aplicación es el siguiente, lo guardaremos en un archivo llamado app.py:

app.py

```python
from flask import Flask
from redis import Redis, RedisError
import os
import socket

# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

app = Flask(__name__)

@app.route("/")
def hello():
    try:
        visits = redis.incr("counter")
    except RedisError:
        visits = "<i>cannot connect to Redis, counter disabled</i>"

    html = "<h3>Hello {name}!</h3>" \
           "<b>Hostname:</b> {hostname}<br/>" \
           "<b>Visits:</b> {visits}"
    return html.format(name=os.getenv("NAME", "world"), hostname=socket.gethostname(),  visits=visits)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)
```

Nuestra aplicación tiene una serie de dependencias (librerías de terceros) que guardaremos en el archivo requirements.txt:

```
app.py        requirements.txt ×
requirements.txt
1   Flask
2   Redis
```

Y por último definimos nuestro Dockerfile:

```
Dockerfile ×
Dockerfile > ...
1   # Partimos de una base oficial de python
2   FROM python:3-slim
3
4   # El directorio de trabajo es desde donde se ejecuta el contenedor al iniciarse
5   WORKDIR /app
6
7   # Copiamos todos los archivos del build context al directorio /app del contenedor
8   COPY . /app
9
10  # Ejecutamos pip para instalar las dependencias en el contenedor
11  RUN pip install --trusted-host pypi.python.org -r requirements.txt
12
13  # Indicamos que este contenedor se comunica por el puerto 80/tcp
14  EXPOSE 80
15
16  # Declaramos una variable de entorno
17  ENV NAME World
18
19  # Ejecuta nuestra aplicación cuando se inicia el contenedor
20  CMD ["python", "app.py"]
```

En total debemos tener 3 archivos:

```
Diplomado/friendlyhello on ?docker [?] on v26.0.0
→ ls
Dockerfile          app.py          requirements.txt
```

Ahora construimos la imagen de nuestra aplicación:

```
docker build -t friendlyhello .
```

```
Diplomado/friendlyhello on ?docker [?] on v26.0.0
→ docker build -t friendlyhello .

[+] Building 11.0s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 636B
 => [internal] load metadata for docker.io/library/python:3-slim
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/python:3-slim@sha256:e3ae8cf03c4f0abbfef13a8147478a7cd92798a94fa729a36a185d9106cbae32
 => => resolve docker.io/library/python:3-slim@sha256:e3ae8cf03c4f0abbfef13a8147478a7cd92798a94fa729a36a185d9106cbae32
 => => sha256:faf6686d57dd22140a03308e5af4f0c962d8eb80e0f10deacd578a8193f4adcf 7.30kB / 7.30kB
 => => sha256:09f376ebb190216b0459f470e71bec7b5dfa611d66bf008492b40dcc5f1d8eae 29.15MB / 29.15MB
 => => sha256:578d3daee00a7de4583e0d1816f57d0c2aac5d1a327088d5e79bf256609560c9 3.51MB / 3.51MB
 => => sha256:2614b8b77544e64061baa8ac85bba068f597ac780d8f06fb1f8d84a29c500aab 12.01MB / 12.01MB
 => => sha256:e3ae8cf03c4f0abbfef13a8147478a7cd92798a94fa729a36a185d9106cbae32 1.65kB / 1.65kB
 => => sha256:d8262afaf6a091994a330d99cb0193d899177c3f2be50228cded1577a327dbde 1.37kB / 1.37kB
 => => sha256:0d69f3ec269e5686cc1444350c67ef37bc1f2cfea9f3ae9727102657e4485e64 243B / 243B
 => => sha256:c9e6b31e1b2e1e8eba0cc99ef412a61e7b7de639739423726945ae8cba1080df 3.02MB / 3.02MB
 => => extracting sha256:09f376ebb190216b0459f470e71bec7b5dfa611d66bf008492b40dcc5f1d8eae
 => => extracting sha256:578d3daee00a7de4583e0d1816f57d0c2aac5d1a327088d5e79bf256609560c9
 => => extracting sha256:2614b8b77544e64061baa8ac85bba068f597ac780d8f06fb1f8d84a29c500aab
 => => extracting sha256:0d69f3ec269e5686cc1444350c67ef37bc1f2cfea9f3ae9727102657e4485e64
 => => extracting sha256:c9e6b31e1b2e1e8eba0cc99ef412a61e7b7de639739423726945ae8cba1080df
 => [internal] load build context
 => => transferring context: 1.40kB
 => [2/4] WORKDIR /app
 => [3/4] COPY . /app
 => [4/4] RUN pip install --trusted-host pypi.python.org -r requirements.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:34fe643129425ed75f87c84feef63cdd7b51a464234fd712dfaf63ccf055e045
 => => naming to docker.io/library/friendlyhello
```

Y comprobamos que está creada:

```
Diplomado/friendlyhello on ?docker [?] on v26.0.0 took 12s
→ docker image ls
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
friendlyhello   latest    34fe64312942   28 seconds ago   148MB
```

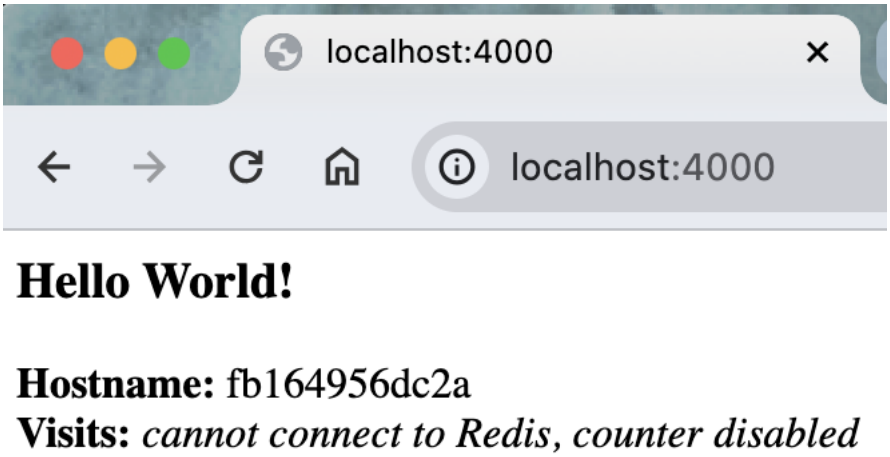
Vamos a arrancar nuestro contenedor y probar la aplicación:

```
docker run --rm -p 4000:80 friendlyhello
```

```
Diplomado/friendlyhello on ?docker [?] on v26.0.0
→ docker run --rm -p 4000:80 friendlyhello

 * Serving Flask app 'app'
 * 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://127.0.0.1:80
 * Running on http://172.17.0.2:80
Press CTRL+C to quit
```

Obtendremos un mensaje como este:



Creando la Aplicación

Vamos a crear el siguiente archivo docker-compose.yaml:

docker-compose.yml

docker-compose.yml

```yaml
services:
    web:
        build: .
        labels:
        - "traefik.enable=true"
        - "traefik.http.routers.web.rule=Host(`localhost`)"
        - "traefik.http.routers.web.entrypoints=web"
        - "traefik.http.services.web.loadbalancer.server.port=80"
    redis:
        image: redis
        volumes:
            - "./data:/data"
        command: redis-server --appendonly yes
        labels:
        - "traefik.enable=false"
    traefik:
        image: traefik:v2.3
        command:
        - "--log.level=DEBUG"
        - "--api.insecure=true"
        - "--providers.docker=true"
        - "--providers.docker.exposedByDefault=false"
        - "--entrypoints.web.address=:4000"
        ports:
        - "4000:4000" # Exponer Traefik en el puerto 4000 de localhost
        - "8080:8080" # Dashboard de Traefik
        volumes:
        - "/var/run/docker.sock:/var/run/docker.sock"
        labels:
        - "traefik.enable=true"
```

Vamos a arrancar esta nueva aplicación, pero esta vez añadiendo varios servicios web:

```
docker compose up -d --scale web=5
```

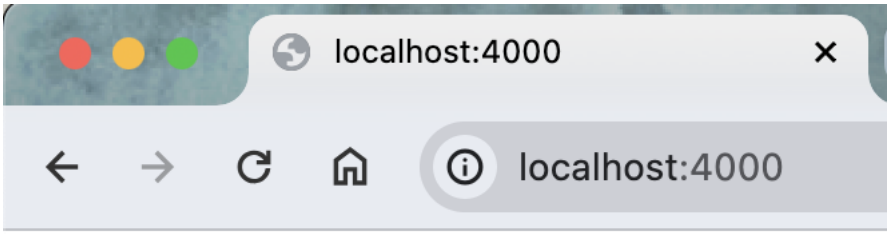
```
Diplomado/friendlyhello on ?docker [?] on v26.0.0
→ docker compose up -d --scale web=5

[+] Running 14/14
 ✔ traefik Pulled
   ✔ 0a6724ff3fcd Pull complete
   ✔ 64d0c2f48fed Pull complete
   ✔ 00390834f324 Pull complete
   ✔ 059f159f3940 Pull complete
 ✔ redis Pulled
   ✔ 09f376ebb190 Already exists
   ✔ 9ae6a7172b01 Pull complete
   ✔ 2c310454138b Pull complete
   ✔ 3eba9ec960aa Pull complete
   ✔ 3d36c165ff0a Pull complete
   ✔ 493d196d734f Pull complete
   ✔ 4f4fb700ef54 Pull complete
   ✔ 484e0560ae90 Pull complete
[+] Building 5.0s (9/9) FINISHED
 => [web internal] load build definition from Dockerfile
 => => transferring dockerfile: 636B
 => [web internal] load metadata for docker.io/library/python:3-slim
 => [web internal] load .dockerignore
 => => transferring context: 2B
 => [web 1/4] FROM docker.io/library/python:3-slim@sha256:e3ae8cf03c4f0abbfef13a8147478a7cd92798a94fa729a36a185d9106cbae32
 => [web internal] load build context
 => => transferring context: 1.08kB
 => CACHED [web 2/4] WORKDIR /app
 => [web 3/4] COPY . /app
 => [web 4/4] RUN pip install --trusted-host pypi.python.org -r requirements.txt
 => [web] exporting to image
 => => exporting layers
 => => writing image sha256:e5b1c47cfabce44ae847442024ad6056a2378e4b28cbaa13b8694cc873cce162
 => => naming to docker.io/library/friendlyhello-web
[+] Running 8/8
 ✔ Network friendlyhello_default      Created
 ✔ Container friendlyhello-web-2      Started
 ✔ Container friendlyhello-web-1      Started
 ✔ Container friendlyhello-web-5      Started
 ✔ Container friendlyhello-web-3      Started
 ✔ Container friendlyhello-redis-1    Started
 ✔ Container friendlyhello-traefik-1  Started
 ✔ Container friendlyhello-web-4      Started
```

usamos docker ps para ver los contenedores disponibles tendremos:

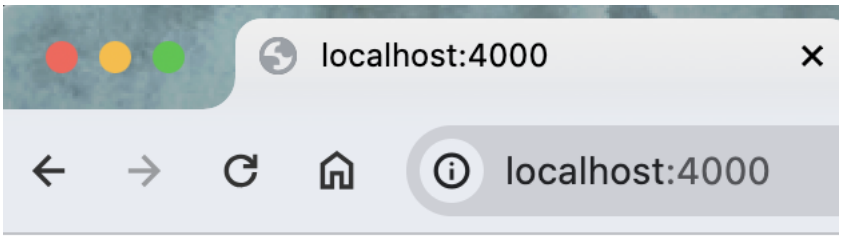
```
Diplomado/friendlyhello on ?docker [?] on v26.0.0 took 13s
→ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                                                    NAMES
7fa6432a2118   redis              "docker-entrypoint.s…"   27 seconds ago   Up 26 seconds   6379/tcp                                                 friendlyhello-redis-1
38fa64f5b180   friendlyhello-web  "python app.py"          27 seconds ago   Up 26 seconds   80/tcp                                                   friendlyhello-web-4
a4e783703fe4   traefik:v2.3       "/entrypoint.sh --lo…"   27 seconds ago   Up 26 seconds   0.0.0.0:4000->4000/tcp, 80/tcp, 0.0.0.0:8080->8080/tcp   friendlyhello-traefik-1
ec8dc870531f   friendlyhello-web  "python app.py"          27 seconds ago   Up 26 seconds   80/tcp                                                   friendlyhello-web-3
62f2749105d1   friendlyhello-web  "python app.py"          27 seconds ago   Up 26 seconds   80/tcp                                                   friendlyhello-web-2
c7fde7e0e4e3   friendlyhello-web  "python app.py"          27 seconds ago   Up 26 seconds   80/tcp                                                   friendlyhello-web-1
c27c046e4614   friendlyhello-web  "python app.py"          27 seconds ago   Up 26 seconds   80/tcp                                                   friendlyhello-web-5
```

En esta ocasión vamos recargando la página, veremos cómo cambian los hostnames, que a su vez coinciden con los identificadores de los contenedores anteriores.



Hello World!

Hostname: 38fa64f5b180
Visits: 6

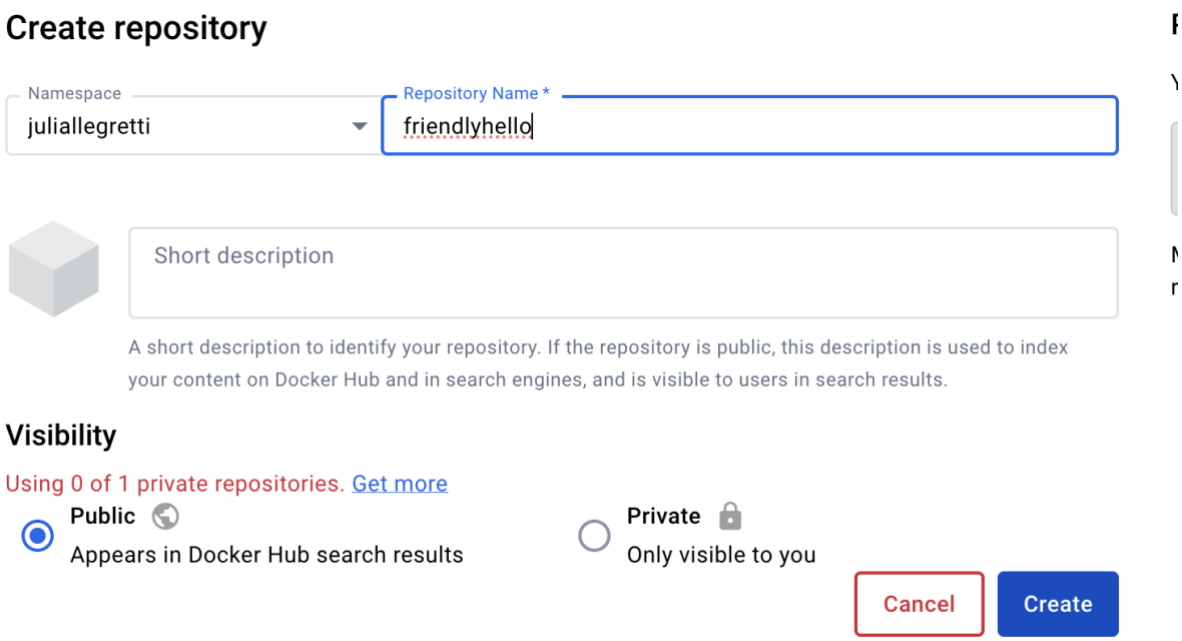


Hello World!

Hostname: 62f2749105d1
Visits: 7

Compartir Imágenes

Creamos un repositorio en DockerHub



Luego realizamos el docker login

```
Diplomado/friendlyhello on docker [?] on v26.0.0
→ docker login
Log in with your Docker ID or email address to push and pull images f
You can log in with your password or a Personal Access Token (PAT). U

Username: juliallegretti
Password:
Login Succeeded
```

Hacemos build de la imagen con nuestro workspace para que se puedan guardar

`docker build -t username/friendlyhello .`

```
Diplomado/friendlyhello on ?docker [?] on v26.0.0 took 3s
→ docker build -t juliallegretti/friendlyhello .
[+] Building 3.7s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 636B
 => [internal] load metadata for docker.io/library/python:3-slim
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/python:3-slim@sha256:e3ae8cf03c4f0abbfef13a8147478a7cd92798a94fa729a36a185d9106cbae32
 => [internal] load build context
 => => transferring context: 449B
 => CACHED [2/4] WORKDIR /app
 => CACHED [3/4] COPY . /app
 => [4/4] RUN pip install --trusted-host pypi.python.org -r requirements.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:ea948a0edfbf27ac8aae3bf66b142accc400881382415e1584901ad1bd6e1c26
 => => naming to docker.io/juliallegretti/friendlyhello
```

Subimos a imagen

```
Diplomado/friendlyhello on ?docker [?] on v26.0.0 took 5s
→ docker push juliallegretti/friendlyhello
Using default tag: latest
The push refers to repository [docker.io/juliallegretti/friendlyhello]
e1fe5f01ce40: Pushed
931ae85addd6: Pushed
e69ab5e13f2c: Pushed
ee05ff71d16f: Mounted from library/python
9e155b490ab8: Mounted from library/python
414ffc85acfb: Mounted from library/python
071988c3d0ae: Mounted from library/python
5d4427064ecc: Mounted from library/redis
latest: digest: sha256:488e31a3491200d027e73a0d61bb2280d4452d7f8f1966cc014a5466f0e713f3 size: 1995
```

Comprobamos

juliallegretti/friendlyhello

Updated less than a minute ago

This repository does not have a description INCOMPLETE

This repository does not have a category INCOMPLETE

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	---	a few seconds ago

See all