

# Práctica 3 - Docker

Prof.: Emma di Battista

Estudiante: Rommel Contreras

Carnet: 14-10242

Para esta práctica trabajará con **el mismo repositorio git** que creó para la práctica anterior.

A su jefe en DeLab le gustó el inventario automatizado que extrae su script, y quiere que el CSV que genera su script este **siempre actualizado**, por lo que le ha pedido que ese script se ejecute en la plataforma docker en producción de la compañía.

1. Modifique el código de inventario de la asignación anterior para que la consulta se ejecute cada 5 minutos, y cada vez que se ejecute, actualice el archivo CSV generado. (Utilice el módulo "time" de python para el temporizador)

Para realizar este inciso se importó la librería *time* de *Python* y se agregó a la sección de imports de nuestro código anterior:

```
1
2 ##### Imports #####
3 from dataclasses import dataclass
4 import pprint
5 import json
6 from urllib import response
7 import requests
8 import csv
9 import os
10 from pprint import pprint
11 import time
12 from time import time, ctime, sleep
13 ##### Utils #####
```

*Ilustración 1 sección de imports de las librerías necesarias para el código DeLab.py*

Luego se creó una función *countdown()* para realizar el contador de los 5 minutos la cual se muestra a continuación:

```
162 def countdown():
163     sec=5*60
164     while sec:
165         sleep(1)
166         sec-=1
167         #print(sec) #Debug print
168
```

*Ilustración 2 función countdown() para contar 5 minutos*

Se cambió la aplicación principal que solicitaba al usuario una entrada y se adecuó a las necesidades del proyecto para que se hicieran las consultas a la API Cada 5 minutos.

```
179 ##### main aplicacion #####
180
181 while(1):
182     orgData=org_Data()
183     #oId=o_id() # si se desea que el código sea dinamico habilitar esta opción de nuevo
184     oId='681155'
185     orgDev= org_Dev(oId)
186     devStatus= deviceStatuses(oId)
187     orgDev=productType()
188     formpt(orgDev,devStatus)
189     jsontocsv(orgDev)
190     pp(orgDev)
191     countdown()
```

*Ilustración 3 Nueva aplicación principal*

2. Escriba un Dockerfile para crear un contenedor que ejecute el código python del punto anterior.

```
Dockerfile > ...
1 FROM python:3.11.0b5-alpine3.15
2 WORKDIR /usr/src/app
3
4
5 COPY requirements.txt ./
6 RUN pip install --no-cache-dir -r requirements.txt
7
8 COPY . .
9
10 EXPOSE 80
11
12 CMD [ "python", "./DeLab.py" ]
13
```

*Ilustración 4 Dockerfile*

3. Construya y ejecute el contenedor desde la línea de comandos.

```

rommelaser@Tokyo:~/Documents/SDN/DeLab$ docker build -t delab-py .
[+] Building 1.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 239B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/python:3.11.                  1.0s
=> [1/5] FROM docker.io/library/python:3.11.0b5-alpine3.15@sha                 0.0s
=> => resolve docker.io/library/python:3.11.0b5-alpine3.15@sha                 0.0s
=> [internal] load build context                                                0.0s
=> => transferring context: 9.94kB                                              0.0s
=> CACHED [2/5] WORKDIR /usr/src/app                                           0.0s
=> CACHED [3/5] COPY requirements.txt ./                                         0.0s
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements                 0.0s
=> [5/5] COPY . .                                                                0.1s
=> exporting to image                                                           0.0s
=> => exporting layers                                                           0.0s
=> => writing image sha256:d8566a854b1aa5446f62702d5d324c955a9                0.0s
=> => naming to docker.io/library/delab-py                                     0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabili
ties and learn how to fix them

```

*Ilustración 5 Construcción de la imagen delab-py*

5. Haga un nuevo commit de su código añadiendo el Dockerfile en el repositorio que creó en la asignación anterior.

```

rommelaser@Tokyo:~/Documents/SDN/DeLab$ git add .
rommelaser@Tokyo:~/Documents/SDN/DeLab$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   DeLab.py
    new file:   Dockerfile
    modified:   requirements.txt

rommelaser@Tokyo:~/Documents/SDN/DeLab$ git commit -m "se añade el Dockerfil
e, se actualizaron los requerimientos (dado a que a que daban problemas para
wsl2 se hizo, python -m pip freeze > requirements.txt desde el powershell d
e windows), se le colcó un tag de fecha a la función que genera el archivo
csv con el objeto de observar cuando el daemon actualizara el archivo"
[master cc33345] se añade el Dockerfile, se actualizaron los requerimientos
(dado a que a que daban problemas para wsl2 se hizo, python -m pip freeze >
requirements.txt desde el powershell de windows), se le colcó un tag de fec
ha a la función que genera el archivo csv con el objeto de observar cuando e
l daemon actualizara el archivo
3 files changed, 17 insertions(+), 2 deletions(-)
create mode 100644 Dockerfile
rewrite requirements.txt (100%)
rommelaser@Tokyo:~/Documents/SDN/DeLab$ git push -u origin/master
error: src refspec u does not match any

```

*Ilustración 6 Commit inciso 5*

6. Cree un volumen para su contenedor, en la dirección donde su código almacena el archivo CSV para hacer que este persista en memoria aún cuando se detenga el contenedor. (Modifique el Dockerfile de ser necesario)

Se puede hacer así:

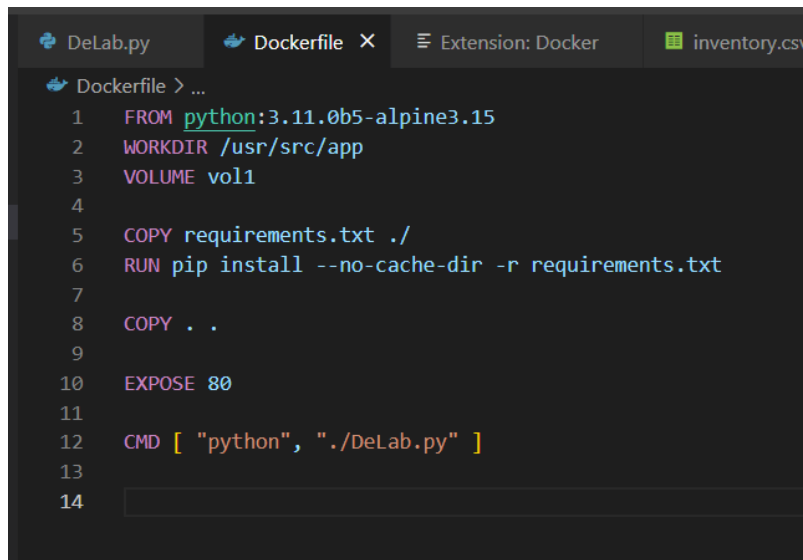
```

rommelaser@Tokyo:~/Documents/SDN/DeLab$ docker run -v vol1:/usr/src/ap
p --name DeLab -itd delab-py
b83e68028bcf2215091417f0a72572a5277074b95a1edd9a023aff3685841658
rommelaser@Tokyo:~/Documents/SDN/DeLab$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS         NAMES
b83e68028bcf   delab-py      "python ./DeLab.py"     6 seconds ago Up 4 s
80/tcp        DeLab
rommelaser@Tokyo:~/Documents/SDN/DeLab$ docker volume ls
DRIVER      VOLUME NAME
local       vol1
rommelaser@Tokyo:~/Documents/SDN/DeLab$ |

```

*Ilustración 7 Creación del volumen y del contenedor incluyendo ejecución de la imagen como demonio con las opciones de interactividad terminal y trabajo en segundo plano (demonio)*

O se puede modificar el Dockerfile



```

DeLab.py Dockerfile X Extension: Docker inventory.csv
Dockerfile > ...
1 FROM python:3.11.0b5-alpine3.15
2 WORKDIR /usr/src/app
3 VOLUME vol1
4
5 COPY requirements.txt ./
6 RUN pip install --no-cache-dir -r requirements.txt
7
8 COPY . .
9
10 EXPOSE 80
11
12 CMD [ "python", "./DeLab.py" ]
13
14

```

*Ilustración 8 Creación del volumen vol1 desde el Dockerfile*

La desventaja de esta es que el volumen queda con un nombre aleatorio

```

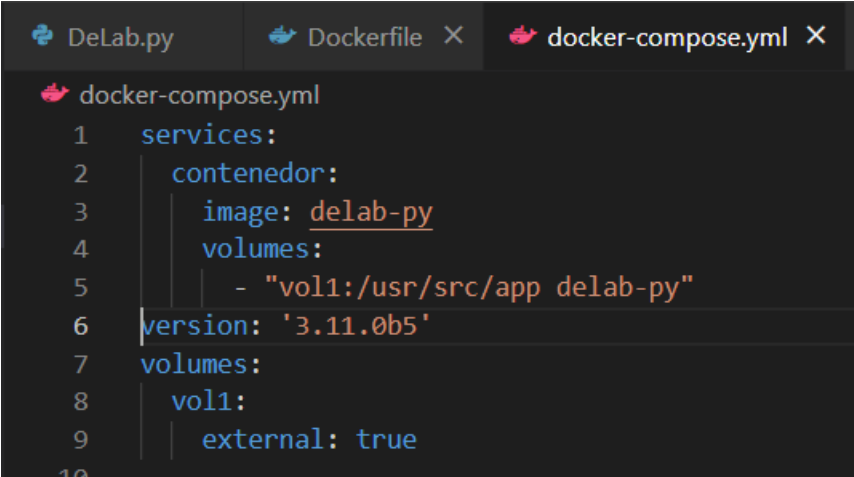
rommelaser@Tokyo:~/Documents/SDN/DeLab$ docker run -itd delab-py
6a0a622c0cda63f9d698504635a414f99a64c719818ce2d4cd584d7f1786fc90
rommelaser@Tokyo:~/Documents/SDN/DeLab$ docker volume ls -a
unknown shorthand flag: 'a' in -a
See 'docker volume ls --help'.
rommelaser@Tokyo:~/Documents/SDN/DeLab$ docker volume ls
DRIVER      VOLUME NAME
local       2140f6c8a10b92a0f491c6e32aca244e13591986a57e1c959a0a3b0e0d67
ebad

```

*Ilustración 9 Ejemplo de que el volumen se crea con un nombre aleatorio*

Por lo que es una mala práctica, así que se usará el otro método y si borrará este volumen.

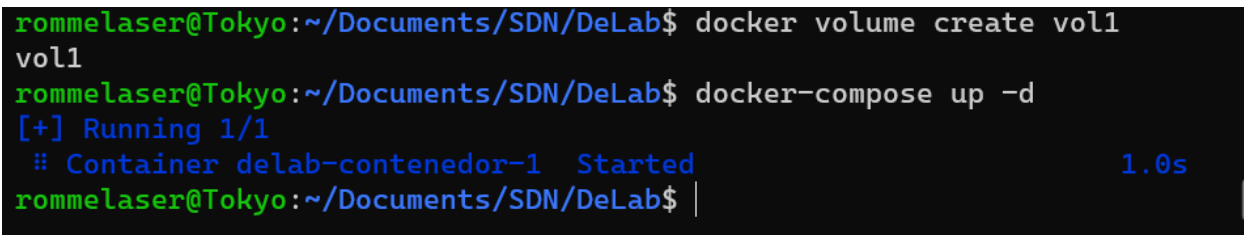
7. Construya un archivo docker-compose.yml a partir del cual se pueda iniciar su contenedor con todos los volúmenes necesarios.



```
1  services:
2    contenedor:
3      image: delab-py
4      volumes:
5        - "vol1:/usr/src/app delab-py"
6  version: '3.11.0b5'
7  volumes:
8    vol1:
9      external: true
```

*Ilustración 10 Archivo docker-compose.yml*

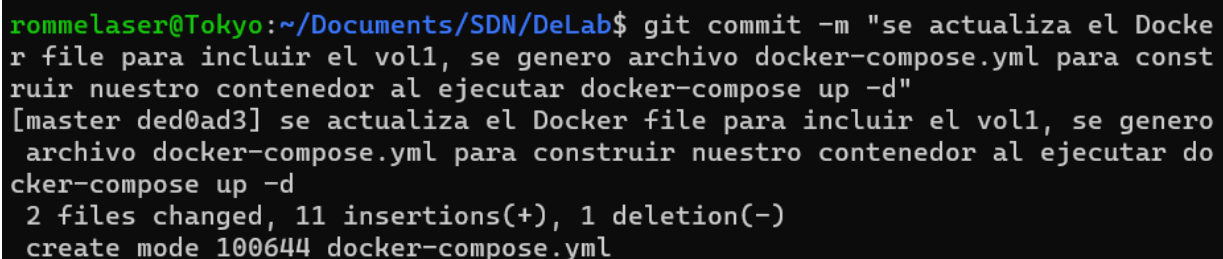
8. Construya y ejecute su contenedor desde la línea de comandos usando el archivo docker compose que creó.



```
rommelaser@Tokyo:~/Documents/SDN/DeLab$ docker volume create vol1
vol1
rommelaser@Tokyo:~/Documents/SDN/DeLab$ docker-compose up -d
[+] Running 1/1
  :: Container delab-contenedor-1 Started 1.0s
rommelaser@Tokyo:~/Documents/SDN/DeLab$ |
```

*Ilustración 11 Iniciado del contenedor mediante docker-compose*

9. Haga commit nuevamente a su repositorio, añadiendo el archivo docker-compose.



```
rommelaser@Tokyo:~/Documents/SDN/DeLab$ git commit -m "se actualiza el Docker
file para incluir el vol1, se genero archivo docker-compose.yml para const
ruir nuestro contenedor al ejecutar docker-compose up -d"
[master ded0ad3] se actualiza el Docker file para incluir el vol1, se genero
archivo docker-compose.yml para construir nuestro contenedor al ejecutar do
cker-compose up -d
2 files changed, 11 insertions(+), 1 deletion(-)
create mode 100644 docker-compose.yml
```

*Ilustración 12 Commit para agregar el docker-compose.yml*