DOCUMENTO DE INSTALACIÓN DEL CHALLENGE DE MERCADO LIBRE

Por: Oscar Bohórquez

Fecha: 19-Mayo-2022

Objetivo:

El presente documento cuenta con las suficientes instrucciones para ejecutar el challenge de forma exitosa y fácil.

Prerequisitos:

1. git cliente o clonar el proyecto desde un navegador
2. Docker instalado de la pagina oficial "https://docs.docker.com/get-docker", instalar la ultima versión de Docker en su Sistema Operativo.

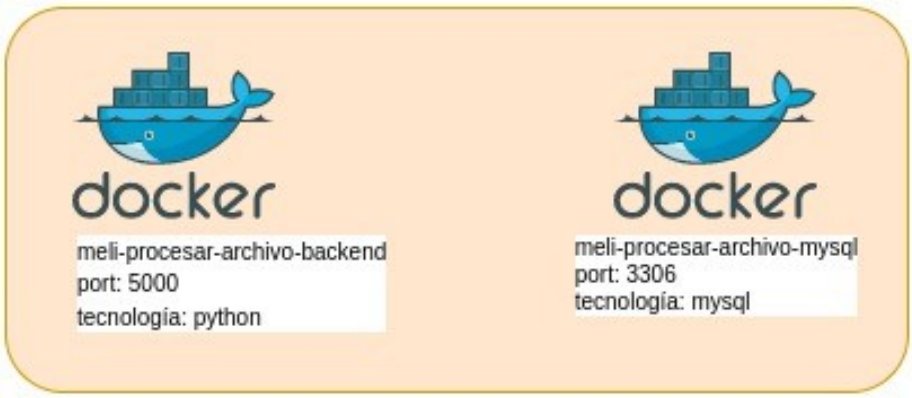
Instalación:

Repositorio con el challenge se encuentra en el siguiente link de github: https://github.com/oscbohr/mvc-python-rest-mysql.git

1. Clonar el repositorio dentro del directorio deseado y entrar al directorio descargado "mvc-python-rest-mysql", revisar que se tengan la siguiente estructura de directorios y archivos:

comando: git clone https://github.com/oscbohr/mvc-python-rest-mysql.git
comando: cd mvc-python-rest-mysql

2. Crear la imagen y ejecutar los contenedores con docker-compose:



NOTA: la aplicación python está ejecutándose por el puerto: 5000 y la base de datos mysql en el puerto 3306; tener en cuenta que si la máquina donde se está ejecutando los contenedores ya tienen esos puertos en uso, debemos cambiarlos en el archivo Dokerfile de cada contenedor al igual que en el archivo “docker-compose.yml”

DockerFile: python

```
# set base image (host OS)
FROM python:3.8.10

# set the working directory in the container
WORKDIR /app

COPY /meli-procesar-archivo/codigo/ /app/

# copy the dependencies file to the working directory
COPY /meli-procesar-archivo/codigo/requirements.txt .

# install dependencies
RUN pip install -r requirements.txt

# copy the all content of directory to the working directory
#COPY . .

RUN ls /app
ENV FLASK_ENV=development
CMD ["python3", "app.py"]
```

docker-compose.yml

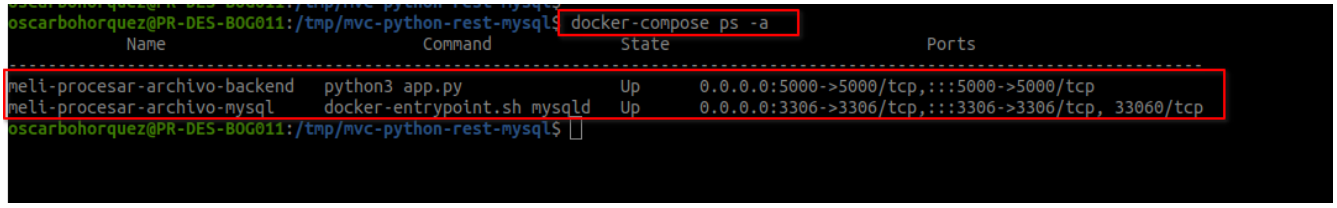
```
version: '3'

services:
  backend:
    build:
      context: .
      dockerfile: ./meli-procesar-archivo/codigo/Dockerfile
    container_name: meli-procesar-archivo-backend
    hostname: backend
    network_mode: bridge
    restart: always
    expose:
      - "5000"
    ports:
      - "5000:5000"
    depends_on:
      - "db"
    links:
      - "db"
  db:
    image: mysql:8.0.29
    network_mode: bridge
    hostname: db
    container_name:  meli-procesar-archivo-mysql
    restart: always
    expose:
      - "3306"
    ports:
      - "3306:3306"
    volumes:
      - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql:ro
    environment:
        - MYSQL_ROOT_PASSWORD=my-secret-pw
        - MYSQL_DATABASE=testDB
```

Ejecutar el comando: “**docker-compose up -d --build**”

```
oscarbohorquez@PR-DES-BOG011:/tmp/mvc-python-rest-mysql$ docker-compose up -d --build
Building backend
Step 1/8 : FROM python:3.8.10
 ---> a369814a9797
Step 2/8 : WORKDIR /app
 ---> Running in 26978a1bd86e
Removing intermediate container 26978a1bd86e
 ---> 2014664b3a75
Step 3/8 : COPY /meli-procesar-archivo/codigo/ /app/
 ---> db4e17fdad58
Step 4/8 : COPY /meli-procesar-archivo/codigo/requirements.txt .
 ---> 86b5d6db12df
Step 5/8 : RUN pip install -r requirements.txt
 ---> Running in 3f0c90370608
Collecting flask==2.1.2
  Downloading Flask-2.1.2-py3-none-any.whl (95 kB)
Collecting Flask-SQLAlchemy==2.5.1
  Downloading Flask_SQLAlchemy-2.5.1-py2.py3-none-any.whl (17 kB)
Collecting requests==2.27.1
  Downloading requests-2.27.1-py2.py3-none-any.whl (63 kB)
Collecting SQLAlchemy==1.4.36
  Downloading SQLAlchemy-1.4.36-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.
Collecting psutil==5.9.0
  Downloading psutil-5.9.0-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.manylinux_2_17_x86_64.m
Collecting pymysql==1.0.2
  Downloading PyMySQL-1.0.2-py3-none-any.whl (43 kB)
Collecting pandas==1.4.2
  Downloading pandas-1.4.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.7 MB)
Collecting cryptography==37.0.2
  Downloading cryptography-37.0.2-cp36-abi3-manylinux_2_24_x86_64.whl (4.0 MB)
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Jinja2>=3.0
```

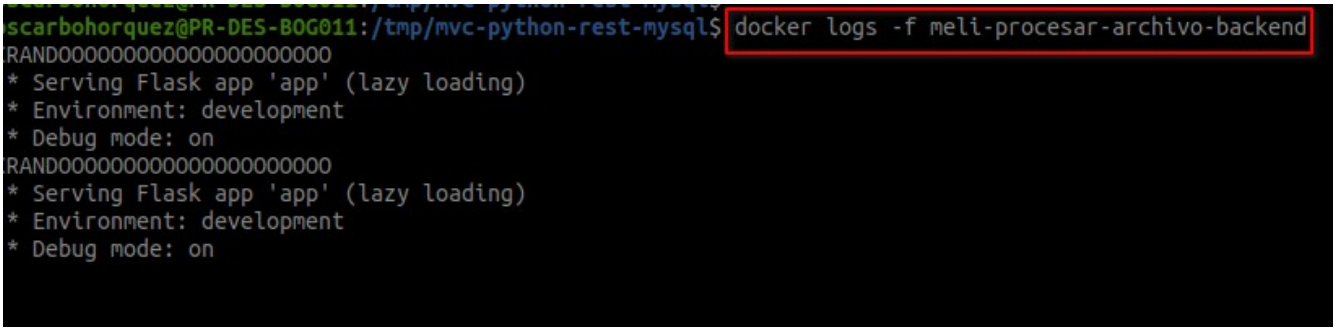
```
Removing intermediate container ba5a63c8508c
 ---> f84d01a26483
Step 7/8 : ENV FLASK_ENV=development
 ---> Running in 33e0d116b344
Removing intermediate container 33e0d116b344
 ---> dff4bb063dc3
Step 8/8 : CMD ["python3", "app.py"]
 ---> Running in b92dc431d295
Removing intermediate container b92dc431d295
 ---> 3b6254d427d7
Successfully built 3b6254d427d7
Successfully tagged mvc-python-rest-mysql_backend:latest
Creating meli-procesar-archivo-mysql ... done
Creating meli-procesar-archivo-backend ... done
oscarbohorquez@PR-DES-BOG011:/tmp/mvc-python-rest-mysql$
```

3. Verificar la correcta ejecución del contenedor de la aplicación backend y de la Base de Datos ejecutando:
   comando: docker-compose ps -a
   resultado:

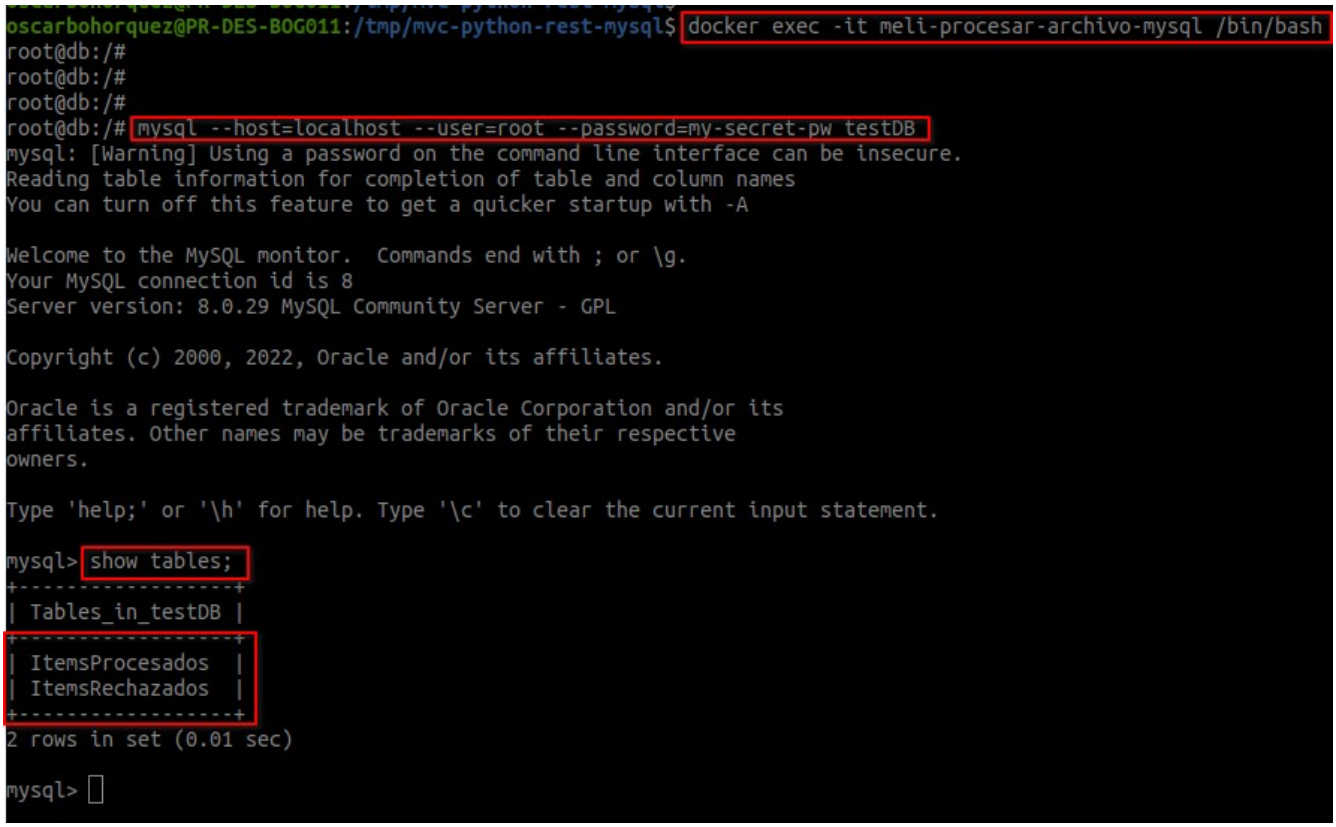


4. Detener e iniciar los contenedores para que tomen el nombre de host configurado en el archivo docker-componse.yml. Para ello ejecute: "docker-compose stop", y seguido "docker-compose start"

5. Verificar los logs de los contenedores ejecutando los siguientes comandos:
   1. "docker logs -f meli-procesar-archivo-backend"
   2. "docker logs -f meli-procesar-archivo-mysql"



6. Ingresar al contenedor de la Base de Datos y verificar la estructura de tablas. Para ello ingrese los siguientes comandos:
   1. “docker exec -it meli-procesar-archivo-mysql /bin/bash”
   2. “mysql --host=localhost --user=root --password=my-secret-pw testDB”
   3. “show tables;”



NOTA: Si la estructura de tablas no se encuentra, ejecutar sobre la misma terminal el script “init.sql” localizado en el directorio “db” del repositorio descargado.

7. Abrir una terminal y ejecutar (en cualquier orden) los siguientes comandos “curl”
   1. POST (Procesar Archivo)

```
curl --header "Content-Type: application/json" --request POST \
 --data '{}' http://localhost:5000/api/v1/procesararchivo
```

```
oscarbohorquez@PR-DES-BOG011:/tmp/mvc-python-rest-mysql$ curl --header "Content-Type: application/json" --request POST \
>  --data '{}' http://localhost:5000/api/v1/procesararchivo
{
  "cantidad": 451,
  "mensaje": "Ok",
  "procesado": true,
  "tiempo": 11.281334400177002
}
oscarbohorquez@PR-DES-BOG011:/tmp/mvc-python-rest-mysql$
```

2. GET (Consultar registros de cargue dada la variable “rows”)

```
curl --header "Content-Type: application/json" --request GET \
 --data '{}' http://localhost:5000/api/v1/procesararchivo?rows=3
```

```
scarbohorquez@PR-DES-BOG011:/tmp/mvc-python-rest-mysql$ curl --header "Content-Type: application/json" --request GET \
  --data '{}' http://localhost:5000/api/v1/procesararchivo?rows=3
{
  "description": "Otros",
  "name": "Peso argentino",
  "nickname": "TRAETULIBRO",
  "price": 3000.0,
  "site": "MLA",
  "siteID": 806707270,
  "start_time": "2019-08-07T23:00:42.000Z"
},
{
  "description": "Pijamas",
  "name": "Peso argentino",
  "nickname": "LAPREFE",
  "price": 1900.0,
  "site": "MLA",
  "siteID": 849015677,
  "start_time": "2020-04-13T21:30:33.000Z"
},
{
  "description": "Displays y LCD",
  "name": "Peso argentino",
  "nickname": "CHSKATE",
  "price": 2900.0,
  "site": "MLA",
  "siteID": 845041373,
  "start_time": "2020-03-22T21:27:57.000Z"
}
scarbohorquez@PR-DES-BOG011:/tmp/mvc-python-rest-mysql$
```

3. DELETE (Eliminar la información de la BD)

```
curl --header "Content-Type: application/json" --request DELETE \
 --data '{}' http://localhost:5000/api/v1/eliminarprocesoarchivo
```

```
oscarbohorquez@PR-DES-BOG011:/tmp/mvc-python-rest-mysql$ curl --header "Content-Type: application/json" --request DELETE \
>  --data '{}' http://localhost:5000/api/v1/eliminarprocesoarchivo
{
  "cantidad": 902,
  "mensaje": "Ok",
  "procesado": true,
  "tiempo": 1.0300323963165283
}
oscarbohorquez@PR-DES-BOG011:/tmp/mvc-python-rest-mysql$
```

8. Opcional: La aplicación cuenta con un archivo de configuración “config.properties” el cual se puede editar y realizar diferentes pruebas de procesamiento. Este archivo contiene la información de:
    1. Conexión a la Base de Datos
    2. URL’s de consumo de Mercado Libre
        - url_primaria: URL inicial de consulta de items de Mercado Libre
        - urls_secundarias: URL’s para consulta de categorias, monedas y vendedors (son dependientes del resultado del consumo de la url_primaria).
    - Cantidad de hilos de procesamiento
        - hilos_primario: Cantidad de hilos a crear para ejecutar la lectura del archivo y consumir la url_primaria.
        - hilos_secundario: Cantidad de hilos a crear para ejecutar las urls_secundarias.
    3. Información del archivo: Donde se especifica la ruta, formato, extensión, separador enconding y bloque de lectura de filas (chunksize)