

Homework 4 - Compose

Exercise 1

Deploy Back End and Front End with Docker Compose.

1. Create a compose.yml file.

2. Define two services: api and fe , using the private registry images:

docker.jala.pro/docker-training/backend:[TAG]

docker.jala.pro/docker-training/frontend:[TAG]

3. Use a user-defined bridge network to allow inter-container communication.

4. Use volumes to persist data of each service

Resolución

Cree una segunda versión de mi código, esto para poder crear una versión mas limpia, y a crear una sección para guardar los logs que tendré de las llamadas que se hagan al Backend.

Y estos serán un volumen que podres revisar más tarde.

1. Hacer un Build de mi nueva configuración de Frontend y Backend para crear imágenes que pueda usar para testar de manera local. Además, pase la bd de mongo atlas a una imagen de mongo, para evitar problemas, al testar, y que sea mucho más fácil de levantar.

```
>> TodoDocker git:(main)
● docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
mi-frontend         1.0         2f7d46a60553     About an hour ago 528MB
mi-backend          1.0         d36ea5a5c7ab     2 hours ago     220MB
mongo               6.0         89def6a7f74a     2 weeks ago     1.03GB
>> TodoDocker git:(main)
```

2. Una vez hice la prueba de que todo estaba funcionando bien, crea la versión con tag para subir al registry, y subi las imágenes.

```
● docker tag mi-backend:1.0 docker.jala.pro/docker-training/backend:rebeca.pereira-v2
>> TodoDocker git:(main)
● docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
mi-frontend         1.0         2f7d46a60553     About an hour ago 528MB
mi-backend          1.0         d36ea5a5c7ab     2 hours ago     220MB
docker.jala.pro/docker-training/backend  rebeca.pereira-v2 d36ea5a5c7ab     2 hours ago     220MB
mongo               6.0         89def6a7f74a     2 weeks ago     1.03GB
>> TodoDocker git:(main)
● docker tag mi-frontend:1.0 docker.jala.pro/docker-training/frontend:rebeca.pereira-v2
>> TodoDocker git:(main)
● docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
mi-frontend         1.0         2f7d46a60553     About an hour ago 528MB
docker.jala.pro/docker-training/frontend  rebeca.pereira-v2 2f7d46a60553     About an hour ago 528MB
mi-backend          1.0         d36ea5a5c7ab     2 hours ago     220MB
docker.jala.pro/docker-training/backend  rebeca.pereira-v2 d36ea5a5c7ab     2 hours ago     220MB
mongo               6.0         89def6a7f74a     2 weeks ago     1.03GB
>> TodoDocker git:(main)
```

```

>> TodoDocker git:(main)
❯ docker push docker.jala.pro/docker-training/backend:rebeca.pereira-v2
The push refers to repository [docker.jala.pro/docker-training/backend]
d2f62848b12d: Layer already exists
0bc4f57ce702: Pushed
254e724d7786: Pushed
c05bc7cdd421: Pushed
a66f4cd4636b: Pushed
997762e48473: Layer already exists
3d3ed10a62fb: Pushed
f778d6c4f6e9: Layer already exists
80b927dfde6b: Pushed
rebeca.pereira-v2: digest: sha256:d36ea5a5c7abe61d38e256e248274fa0667e1c1bc7a4e849bcb5d8b8ff7073e0 size: 856
>> TodoDocker git:(main)
❯ docker push docker.jala.pro/docker-training/frontend:rebeca.pereira-v2
The push refers to repository [docker.jala.pro/docker-training/frontend]
f067be210530: Pushed
dd71dde834b5: Layer already exists
20fe5e24012e: Pushed
87106b8e6717: Pushed
ee98ace327b3: Pushed
25ff2da83641: Layer already exists
f18232174bc9: Layer already exists
f4b624662d6b: Pushed
1e5a4c89cee5: Layer already exists
7eec64511622: Pushed
rebeca.pereira-v2: digest: sha256:2f7d46a60553e9cc45afff09bbc0683f1972ed93871576b27566f778470a39a7 size: 856

```

3. Revise que las imágenes se hubieran subido.

```

>> TodoDocker git:(main)
❯ curl -u 'robot$docker-training+rebeca.pereira:' https://docker.jala.pro/v2/docker-training/backend/tags/list
{"name":"docker-training/backend","tags":["calebespinoza","estherpadilla","gabriela","grettarocha","jessicaorihuela","lauramontano","rebeca.pereira","rebeca.pereira-v2","rolandovillca","watsonluis"]}
>> TodoDocker git:(main)
❯ curl -u 'robot$docker-training+rebeca.pereira:' https://docker.jala.pro/v2/docker-training/frontend/tags/list
{"name":"docker-training/frontend","tags":["calebespinoza","estherpadilla","gabriela","grettarocha","jessicaorihuela","lauramontano","rebeca.pereira","rebeca.pereira-v2","rolandovillca","watsonluis"]}

```

4. Una vez hecho eso, cree el Docker Compose para que use, las imágenes y tanto del proyecto de Frontend y Backend como la de MongoDB y el bridge network y los volúmenes.
Además, en lugar de manejar archivos '.env,' hice que el Compose directamente enviara los datos, para que sea más fácil de manejar.

```
1 services:
2   mongodb:
3     image: mongo:6.0
4     networks:
5       - app-network
6     volumes:
7       - mongodb-data:/data/db
8     environment:
9       MONGO_INITDB_ROOT_USERNAME: admin
10      MONGO_INITDB_ROOT_PASSWORD: password
11
12   backend:
13     image: docker.jala.pro/docker-training/backend:rebeca.pereira-v2
14     networks:
15       - app-network
16     volumes:
17       - backend-logs:/app/data
18     environment:
19       MONGO_URL: "mongodb://admin:password@mongodb:27017/tasksdb?authSource=admin"
20       FRONTEND_URL: "http://frontend:4173,http://localhost:4173"
21     depends_on:
22       - mongodb
23     ports:
24       - "8000:8000"
25
26   frontend:
27     image: docker.jala.pro/docker-training/frontend:rebeca.pereira-v2
28     networks:
29       - app-network
30     volumes:
31       - frontend-config:/app/dist/src/configs
32     depends_on:
33       - backend
34     ports:
35       - "4173:4173"
36
37 networks:
38   app-network:
39     driver: bridge
40
41 volumes:
42   mongodb-data:
43   backend-logs:
44   frontend-config:
45 // Code here
```

5. Con eso pude iniciar el Compose, y como se puede ver en la imagen, se esta haciendo Pull de las imágenes de Frontend, Backend y mongo

```
>> TodoDocker git:(main)
o docker compose up --build
[+] Running 25/28
- mongodb [#####] 240.1MB / 261.2MB Pulling 35.0s
  ✓ e8f7cd07b86a Download complete 0.7s
  ✓ 04dfe4517ae9 Download complete 1.6s
  ✓ 1f8e956554f2 Download complete 0.8s
  - 007fd3d08b1b Downloading [=====] 209.7MB/230.2MB 32.2s
  ✓ c1b4a1e8b877 Download complete 1.7s
  ✓ 215ed5a63843 Download complete 14.9s
  ✓ f2073f89ae4c Download complete 1.0s
  ✓ 212ee3b71c0e Download complete 2.2s
  ✓ backend Pulled 28.8s
  ✓ 3d3ed10a62fb Download complete 0.3s
  ✓ 997762e48473 Download complete 0.2s
  ✓ 0bc4f57ce702 Download complete 6.6s
  ✓ f778d6c4f6e9 Download complete 1.7s
  ✓ c05bc7cdd421 Download complete 0.3s
  ✓ 254e724d7786 Download complete 17.9s
  ✓ d2f62848b12d Download complete 9.3s
  ✓ a66f4ed4636b Download complete 0.2s
- frontend [#####] 97.18MB / 98.01MB Pulling 35.0s
  ✓ 25ff2da83641 Download complete 0.4s
  ✓ f18232174bc9 Download complete 1.8s
  ✓ 20fe5e24012e Download complete 0.2s
  ✓ f4b624662d6b Download complete 0.4s
  ✓ 1e5a4c89cee5 Download complete 0.4s
  ✓ ee98ace327b3 Download complete 12.4s
  ✓ 7eec64511622 Download complete 16.0s
  ✓ dd71dde834b5 Download complete 20.2s
  ✓ f067be210530 Download complete 0.6s

[+] Running 7/7
  ✓ Network tododocker_app-network Created 0.2s
  ✓ Volume "tododocker_frontend-config" Created 0.0s
  ✓ Volume "tododocker_mongodb-data" Created 0.0s
  ✓ Volume "tododocker_backend-logs" Created 0.0s
  ✓ Container tododocker-mongodb-1 Created 3.3s
  ✓ Container tododocker-backend-1 Created 0.3s
  ✓ Container tododocker-frontend-1 Created 0.3s
```

6. Si revisamos los logs, podremos ver todas las llamadas hechas

```
>> TodoDocker git:(main)
o docker compose logs backend
backend-1 | /root/.local/lib/python3.13/site-packages/pydantic/_internal/_config.py:373: UserWarning: Valid config keys have c
backend-1 | hanged in V2:
backend-1 | | * 'allow_population_by_field_name' has been renamed to 'validate_by_name'
backend-1 | | * 'orm_mode' has been renamed to 'from_attributes'
backend-1 | | warnings.warn(message, UserWarning)
backend-1 | Origenes permitidos (CORS): ['http://frontend:4173', 'http://localhost:4173']
backend-1 | INFO: Started server process [1]
backend-1 | INFO: Waiting for application startup.
backend-1 | INFO: Application startup complete.
backend-1 | INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
backend-1 | INFO: 172.18.0.1:55390 - "GET /api/tasks HTTP/1.1" 200 OK
backend-1 | INFO: 172.18.0.1:47664 - "OPTIONS /api/tasks HTTP/1.1" 200 OK
backend-1 | {'_id': ObjectId('682ba1b54f8b21c527639d42'), 'id': None, 'title': 'Ejemplo de tarea 1', 'description': 'Tarea cre
ada con imagen del registry sin completar', 'completed': False}
backend-1 | INFO: 172.18.0.1:47664 - "POST /api/tasks HTTP/1.1" 200 OK
backend-1 | INFO: 172.18.0.1:51832 - "OPTIONS /api/tasks/682ba1b54f8b21c527639d42 HTTP/1.1" 200 OK
backend-1 | INFO: 172.18.0.1:51832 - "PUT /api/tasks/682ba1b54f8b21c527639d42 HTTP/1.1" 200 OK
backend-1 | {'_id': ObjectId('682ba1d74f8b21c527639d43'), 'id': None, 'title': 'Ejemplo de tarea 2', 'description': 'Tarea cre
ada con imagen del registry para probar get all', 'completed': False}
backend-1 | INFO: 172.18.0.1:60360 - "POST /api/tasks HTTP/1.1" 200 OK
backend-1 | {'_id': ObjectId('682ba1e74f8b21c527639d44'), 'id': None, 'title': 'Ejemplo de tarea 3', 'description': 'Para prob
ar el delete', 'completed': True}
backend-1 | INFO: 172.18.0.1:49176 - "POST /api/tasks HTTP/1.1" 200 OK
backend-1 | INFO: 172.18.0.1:49178 - "OPTIONS /api/tasks/682ba1e74f8b21c527639d44 HTTP/1.1" 200 OK
backend-1 | INFO: 172.18.0.1:49178 - "DELETE /api/tasks/682ba1e74f8b21c527639d44 HTTP/1.1" 200 OK
backend-1 | INFO: 172.18.0.1:35160 - "GET /info HTTP/1.1" 200 OK
backend-1 | INFO: 172.18.0.1:44986 - "GET /api/tasks HTTP/1.1" 200 OK
```

7. Si revisamos los logs, que configuramos para guardarse en el app/data, también están funcionando

```
>> TodoDocker git:(main)
o docker compose exec backend cat /app/data/logs/api.log
2025-05-19 21:23:26,764 - INFO - GET /api/tasks - Status: 200
2025-05-19 21:25:09,692 - INFO - POST /api/tasks - Status: 200
2025-05-19 21:25:20,217 - INFO - PUT /api/tasks/682ba1b54f8b21c527639d42 - Status: 200
2025-05-19 21:25:43,972 - INFO - POST /api/tasks - Status: 200
2025-05-19 21:25:59,316 - INFO - POST /api/tasks - Status: 200
2025-05-19 21:26:05,868 - INFO - DELETE /api/tasks/682ba1e74f8b21c527639d44 - Status: 200
2025-05-19 21:26:12,004 - INFO - GET /info - Status: 200
2025-05-19 21:26:29,747 - INFO - GET /api/tasks - Status: 200
```

8. Si revisamos persistencia de datos, con mongo, también funcioanndo.

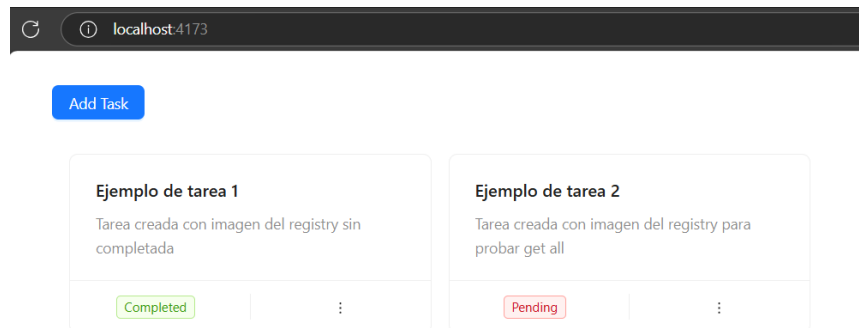
```
test> show dbs
admin          100.00 KiB
config         108.00 KiB
local          72.00 KiB
taskdatabase   72.00 KiB
test> use taskdatabase
switched to db taskdatabase
taskdatabase> show collections
tasks
taskdatabase> db.tasks.find()
[
  {
    _id: ObjectId('682ba1b54f8b21c527639d42'),
    id: null,
    title: 'Ejemplo de tarea 1',
    description: 'Tarea creada con imagen del registry sin completada',
    completed: true
  },
  {
    _id: ObjectId('682ba1d74f8b21c527639d43'),
    id: null,
    title: 'Ejemplo de tarea 2',
    description: 'Tarea creada con imagen del registry para probar get all',
    completed: false
  }
]
```

9. Revisamos conexión de backend y Frontend, mediante curl, todo bien.

```
>> TodoDocker git:(main)
$ docker compose exec frontend curl -s http://backend:8000/info
{"hostname":"a02247db3655","ip":"172.18.0.3","message":"¡Endpoint de información del contenedor!"}
```

10. Y por último si la aplicación está corriendo, todo bien.

- Tasks



- Info

