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.

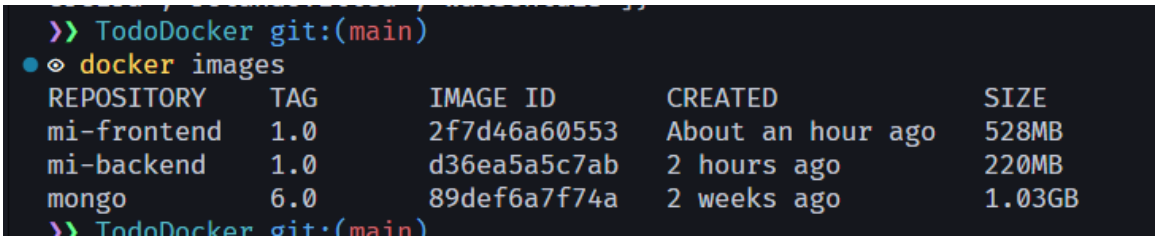
4. Use volumes to persist data of each service

Resolución

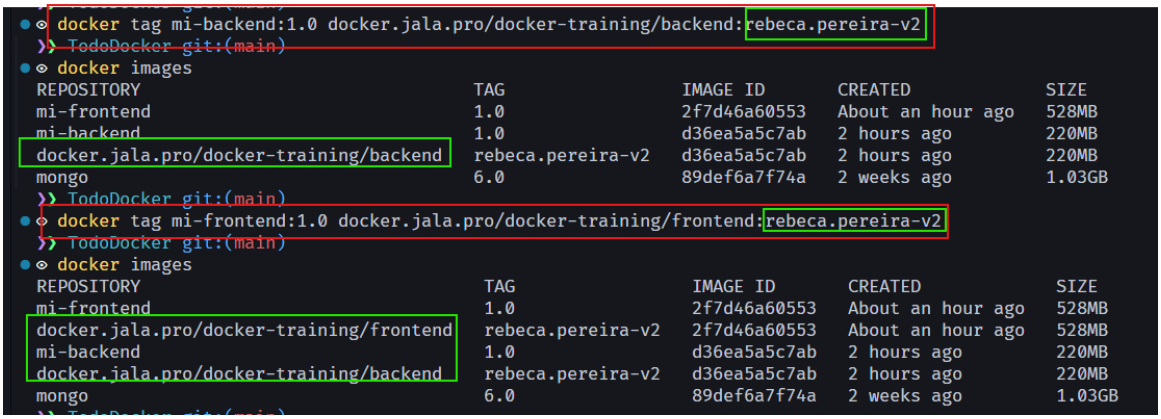
Cree una segunda versión de mi código, esto para poder crear una versión mas limpia, y a crear una sección para guardar los logs que tendré de las llamadas que se hagan al Backend.

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.



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.



```
❯❯ 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.

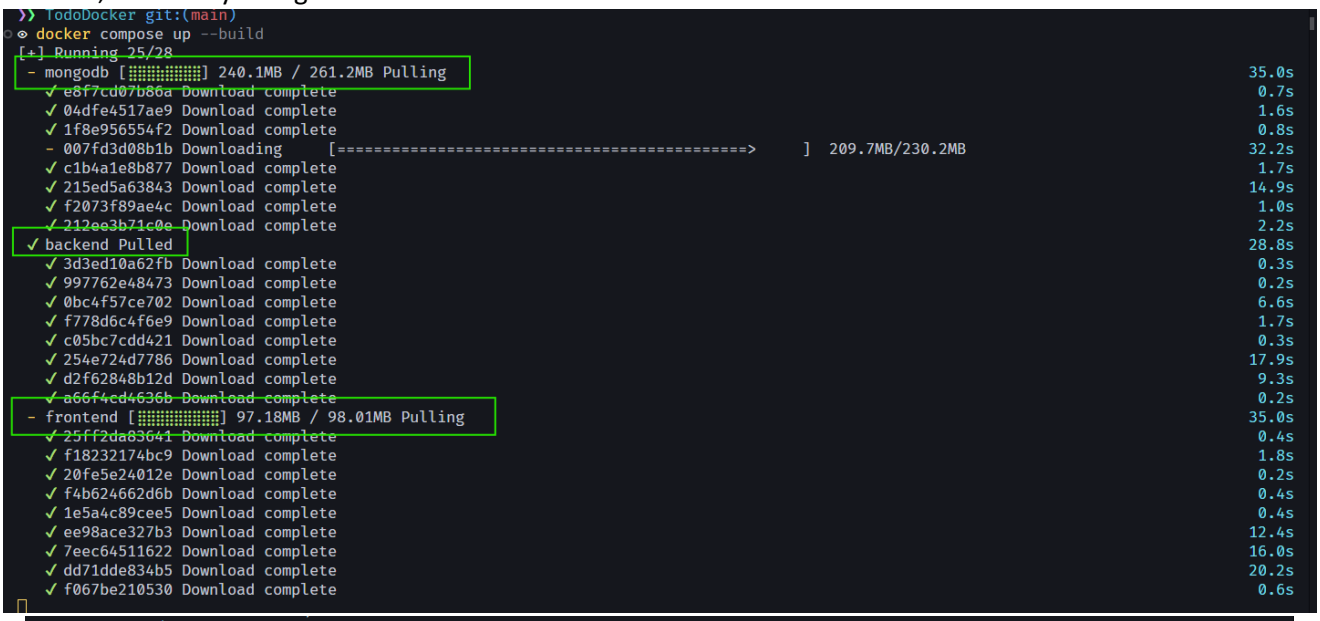
```
❯❯ TodoDocker git:(main)
● ⊙ curl -u 'robot$docker-training+rebeca.pereira:[redacted]' https://docker.jala.pro/v2/docker-training/backend/tags/lis
t
{"name":"docker-training/backend","tags":["calebespinoza","estherpadilla","gabriela","grettarocha","jessicaorihuela","lauramontano","rebeca.p
ereira","rebeca.pereira-v2","rolandovillca","watsonluis"]}
❯❯ TodoDocker git:(main)
● ⊙ curl -u 'robot$docker-training+rebeca.pereira:[redacted]' https://docker.jala.pro/v2/docker-training/frontend/tags/li
st
{"name":"docker-training/frontend","tags":["calebespinoza","estherpadilla","gabriela","grettarocha","jessicaorihuela","lauramontano","rebeca.
pereira","rebeca.pereira-v2","rolandovillca","watsonluis"]}
```

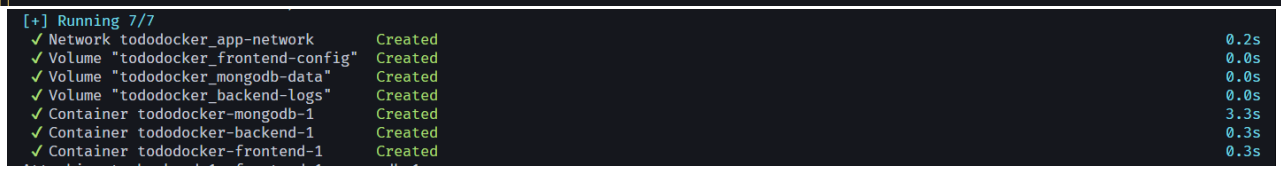
4. Una vez hecho eso, cree el Docker Compose para que use, las imágenes y tanto del proyecto de Frontend y Backend como la de Mongodb y el bridge network y los volumenes.
   Además, en lugar de manejar archivos '.env,' hice que el Compose directamente enviara los datos, para que sea más fácil de manejar.

docker-compose.yml

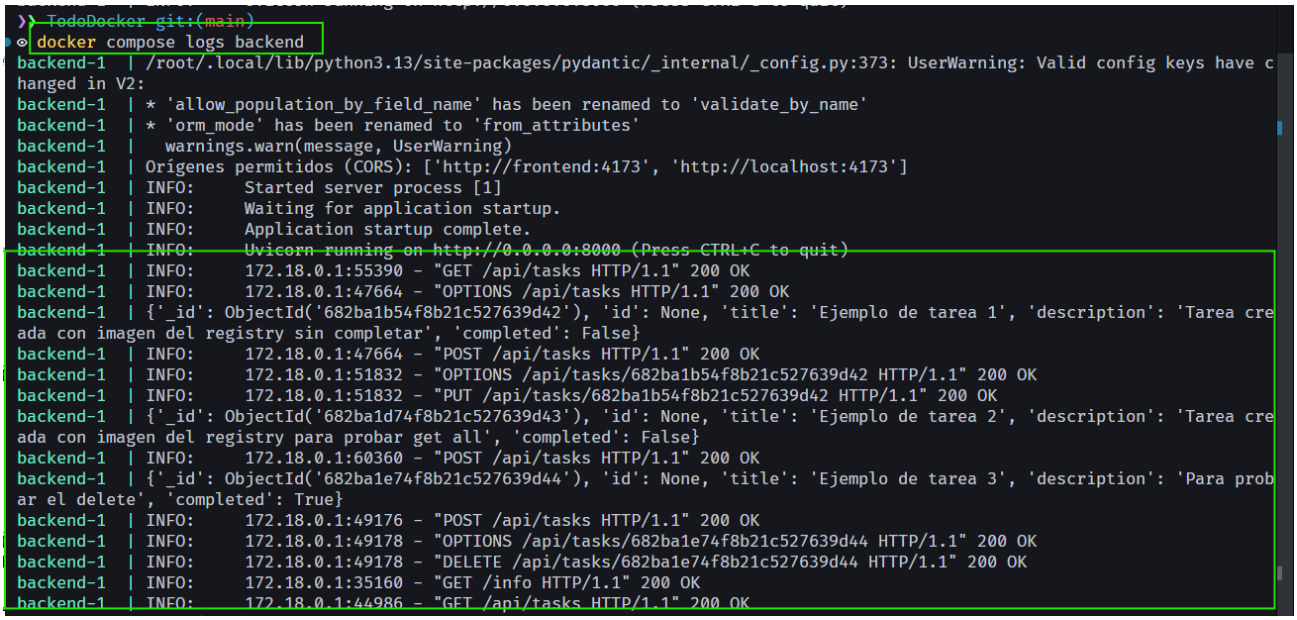
```yaml
services:
  mongodb:
    image: mongo:6.0
    networks:
      - app-network
    volumes:
      - mongodb-data:/data/db
    environment:
      MONGO_INITDB_ROOT_USERNAME: admin
      MONGO_INITDB_ROOT_PASSWORD: password

  backend:
    image: docker.jala.pro/docker-training/backend:rebeca.pereira-v2
    networks:
      - app-network
    volumes:
      - backend-logs:/app/data
    environment:
      MONGO_URL: "mongodb://admin:password@mongodb:27017/tasksdb?authSource=admin"
      FRONTEND_URL: "http://frontend:4173,http://localhost:4173"
    depends_on:
      - mongodb
    ports:
      - "8000:8000"

  frontend:
    image: docker.jala.pro/docker-training/frontend:rebeca.pereira-v2
    networks:
      - app-network
    volumes:
      - frontend-config:/app/dist/src/configs
    depends_on:
      - backend
    ports:
      - "4173:4173"

networks:
  app-network:
    driver: bridge

volumes:
  mongodb-data:
  backend-logs:
  frontend-config:
// 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

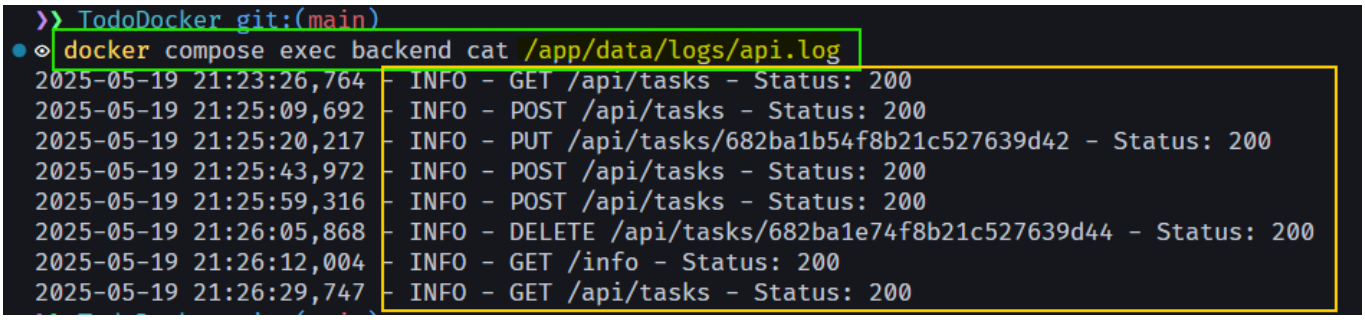




6. Si revisamos los logs, podremos ver todas las llamadas hechas



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



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

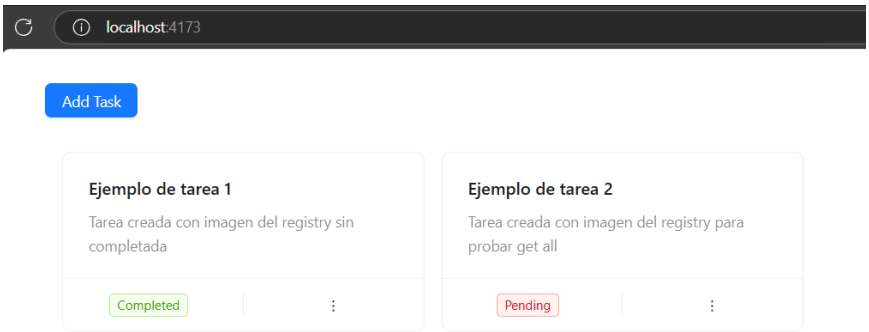
```
test> show dbs
admin         100.00 KiB
config        108.00 KiB
local          72.00 KiB
taskdatabase   72.00 KiB
test> use taskdatabase
switched to db taskdatabase
taskdatabase> show collections
tasks
taskdatabase> db.tasks.find()
[
  {
    _id: ObjectId('682ba1b54f8b21c527639d42'),
    id: null,
    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