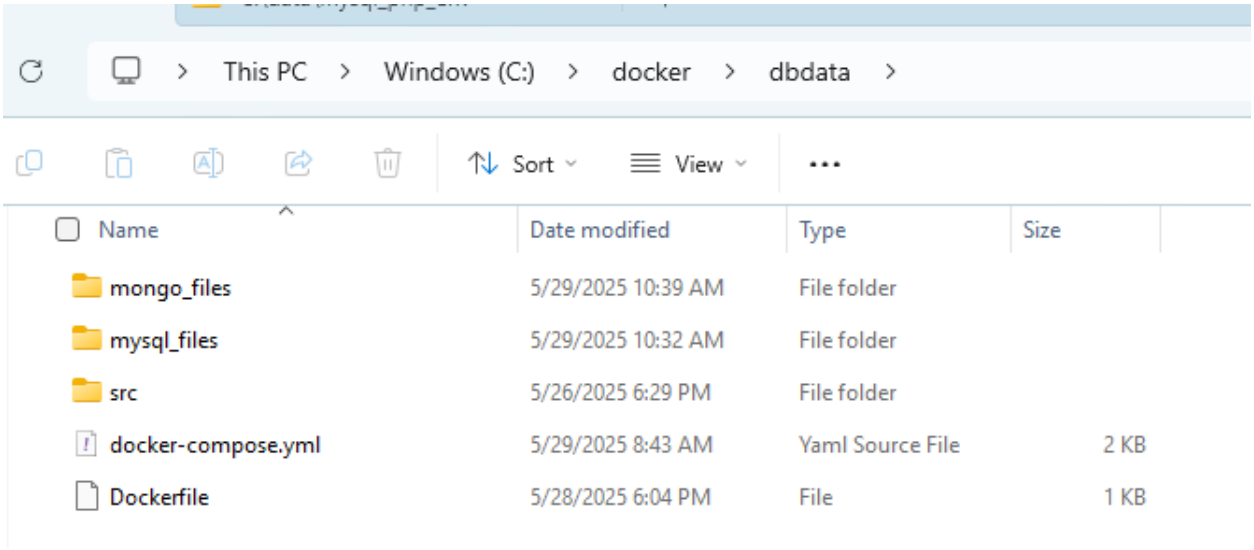
Utilização de Myql e Mongo com Docker

Instalar o Docker Desktop

https://www.docker.com/products/docker-desktop/

Descarregar ficheiro zipado e descomprimi-lo debaixo da pasta c:\docker

Criar a seguinte estrutura de pastas debaixo da pasta que armazena o docker
(c:\docker\dbdata no exemplo)



As pastas mongo_files e mysql_files são opcionais e armazenam ficheiros no disco local que, assim que são ali colocados, são automaticamente copiados (sincronizados) para as pastas internas do Docker (para os containers do mongo e mysql respetivamente). A pasta scr é onde podemos colocar ficheiros php que são lidos pelo apache (equivale a pasta htdocs do XAMPP). Os ficheiros Dockerfile e docker-compose.yml contém a informação para criar os containers. Depois do Docker estar criado vão aparecer mais duas pastas: Mysql_data e mongo_data onde são armazenadas as bases de dados Mysql e mongodb.

Breve explicação do ficheiro docker-compose.yml (fragmento do conteúdo na imagem)

```yaml
  mysql:
    image: mysql:latest
    restart: no
    environment:
      MYSQL_ROOT_PASSWORD: 'root'
      MYSQL_DEFAULT_AUTHENTICATION_PLUGIN: m
    volumes:
      - ./mysql_data/:/var/lib/mysql
      - ./mysql_files:/var/lib/mysql-files/
    ports:
      - "3306:3306"

  phpmyadmin:
    image: phpmyadmin:latest
    restart: always
    ports:
      - 9001:80
    environment:
      PMA_ARBITRARY: 1
    depends_on:
      - mysql

  mongodb:
    image: mongo:latest
    restart: always
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: root
    volumes:
      - ./mongo_data:/data/db
      - ./mongo_files:/tmp/import
```

Cada Serviço é um container, consideram-se no exemplo 4 containers: php, Mysql, Mongo e PhpMyadmin. DepensdsOn indica que o serviço apenas pode “arrancar” depois de os outros arrancarem. Ou seja, por exemplo, o phpmyadmin só pode arrancar depois do Mysql.

Os volumes são importantes pois fazem o mapeamento entre o disco local e a estrutura de ficheiros interna. Por exemplo, a pasta Mysql_data local (debaixo de dbdata) sincroniza com a pasta /var/lib/mysql que está dentro do countainer. Ou seja, sempre que informação é escrita em /var/lib/Mysql (pasta onde seriam guardadas as bases de dados) ela é movida para a pasta Mysql_data do disco local. Não teria de ser assim (aliás esta solução não é recomendada em ambiente de produção), poderia ter sido (em vez de - ./Mysql_data: /var/lib/mysql

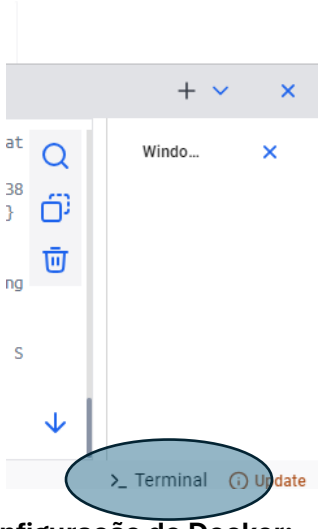
```
  - Mysql_data: /var/lib/mysql
volumes:
Mysql_data:
```

Neste cenário os dados são guardados internamente na pasta `/var/lib/Mysql` dentro do container, o que é mais seguro.

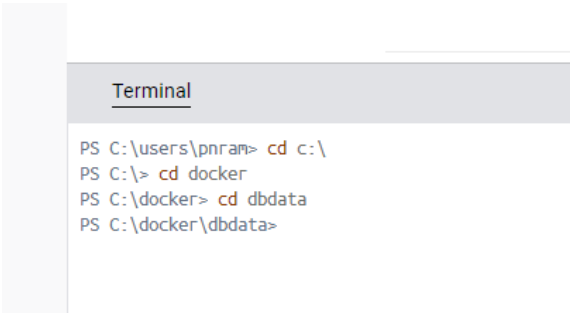
A pasta Mysql_files local (debaixo de dbdata) sincroniza com a pasta /var/lib/Mysql-files, que é a pasta onde por omissão o countainer vai procurar scripts de Mysql.

O resto do ficheiro é autoexplicativo (senhas, portos, nome de pastas, indicação para reinício automático em caso de falha, etc.).

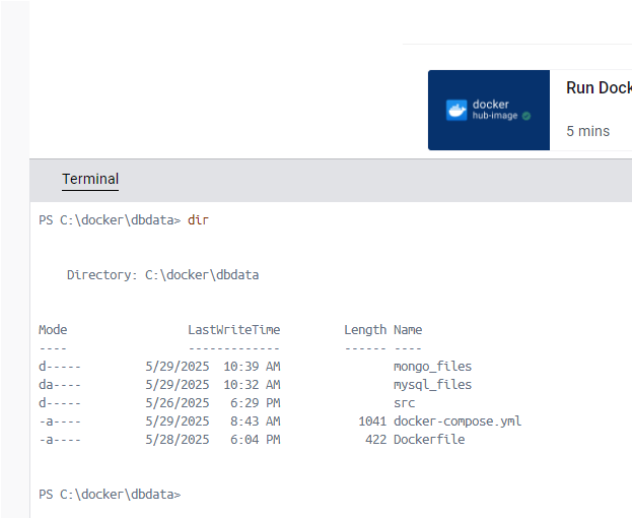
Para abrir o terminal e executar comandos no Docker:



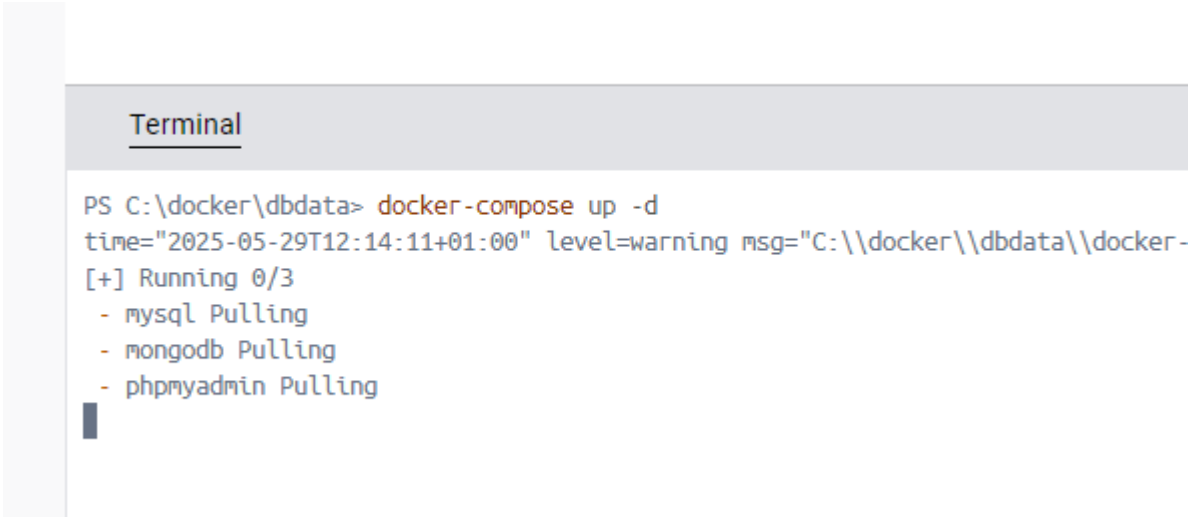
Navegar até à pasta onde está a configuração do Docker:

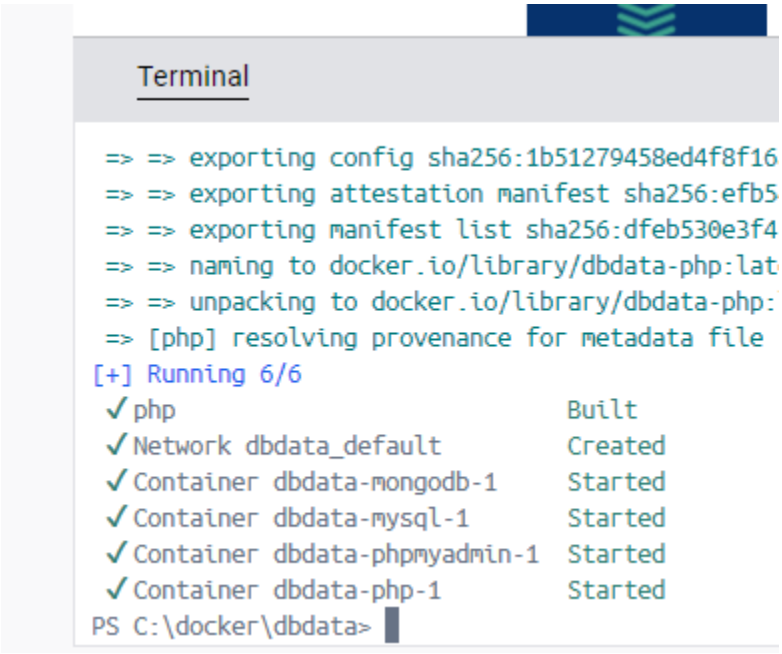


Confirmar que a pasta está completa:



Instalar o Docker (docker-compose up -d**)**

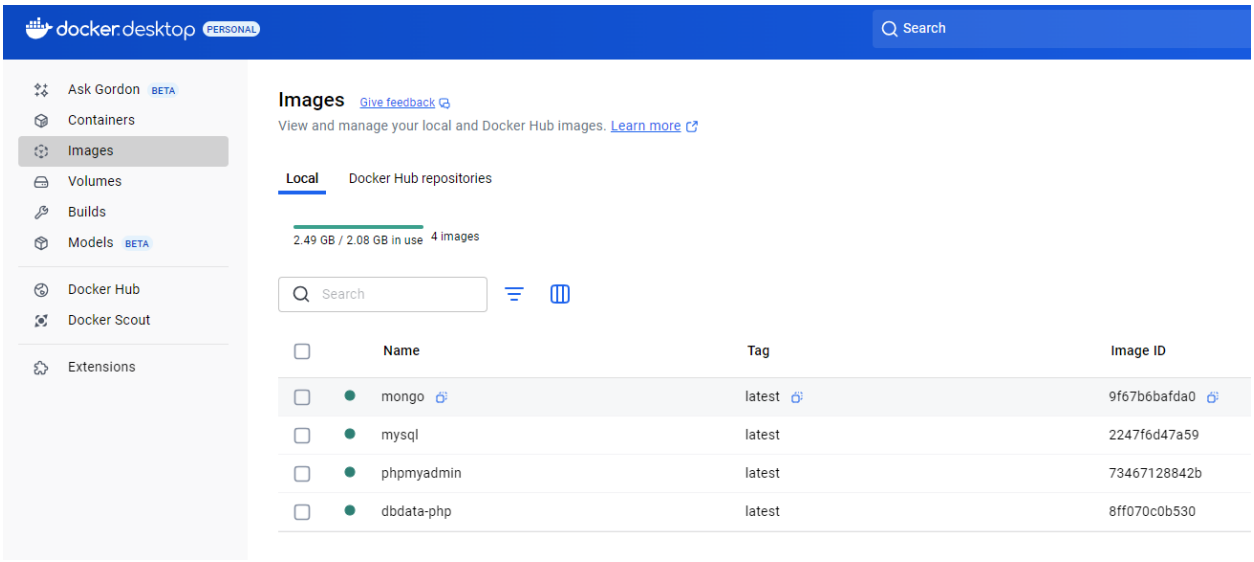




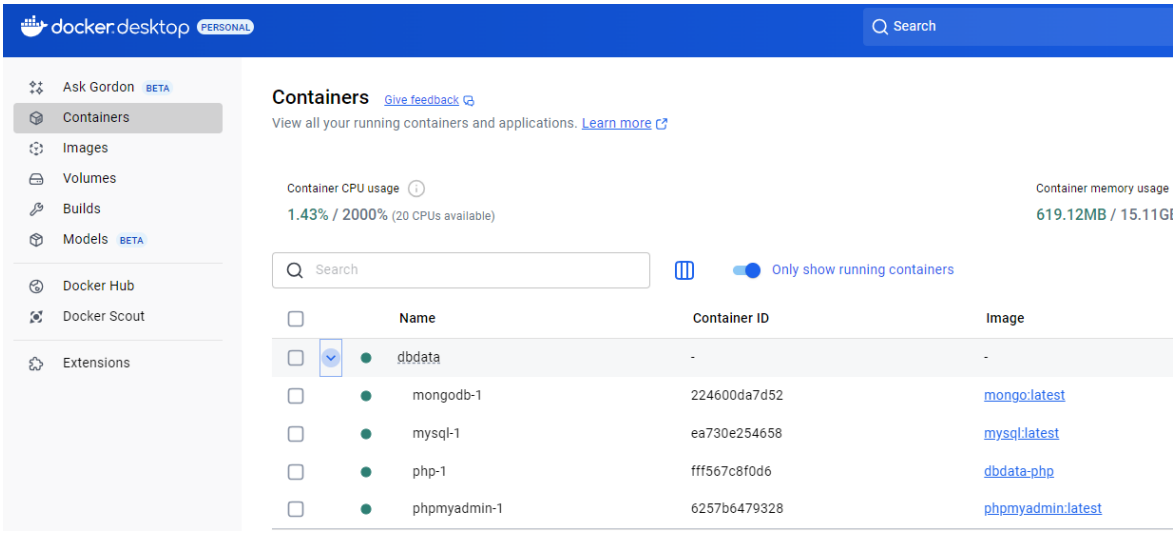
O terminal pode ser o normal do Windows, não tem de ser aberto dentro do Docker. É o mesmo terminal.

Depois de "instalada" a pasta dbdata (docker-compose up**)**

Foram Criadas 4 imagens, equivale aos 4 serviços. A imagem é uma espécie de modelo com informação para executar uma aplicação criada a partir de um dockerfile



E 4 containers, que são as instâncias executáveis das imagens. Enquanto as imagens são apenas de leitura, num container são efetuadas operações de escrita (ficam lá armazenados os dados e ficheiros. Eles são criados, iniciados, parados, reiniciados e destruídos. Se um container estiver com problemas ele pode ser criado a partir da mesma imagem, mas os dados são perdidos (daí ser fundamental as pastas Mysql_data e mongo_data no computador local.

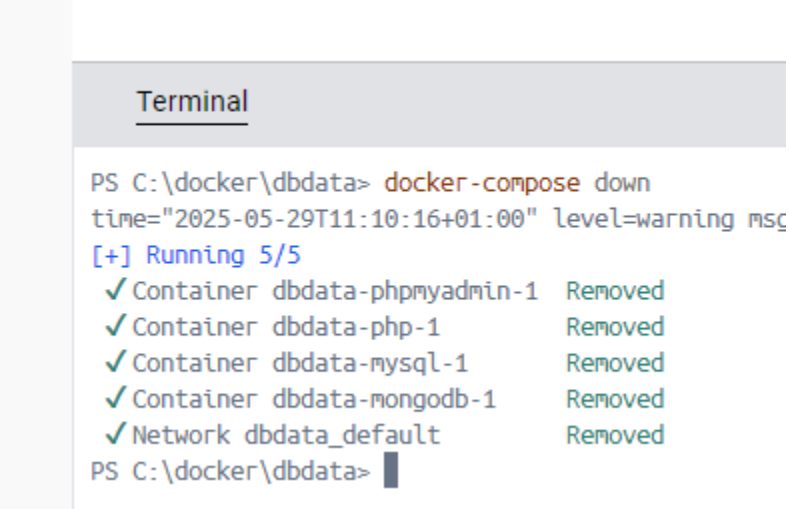


As designações, nomeadamente a componente "-1" são adicionadas automaticamente quando eles são criados pelo comando docker-compose up ou docker run <nome_da_imagem> (por exemplo, docker run Mysql). Para uma mesma imagem podem ser criados vários containers (dai o "-1"). Atenção que o nome do container contém o nome da pasta: dbdata.Mysql-1, por exemplo.

O comando docker-compose up lê o ficheiro docker-compose.yml e para cada serviço cria o container se ele não existir ainda (se já existir volta a criar caso tenha mudado alguma coisa na configuração). Com opção build (docker-compose up –build) ele força a reconstrução das imagens. Com opção d (docker-compose up –d) ele corre em background e liberta o terminal.

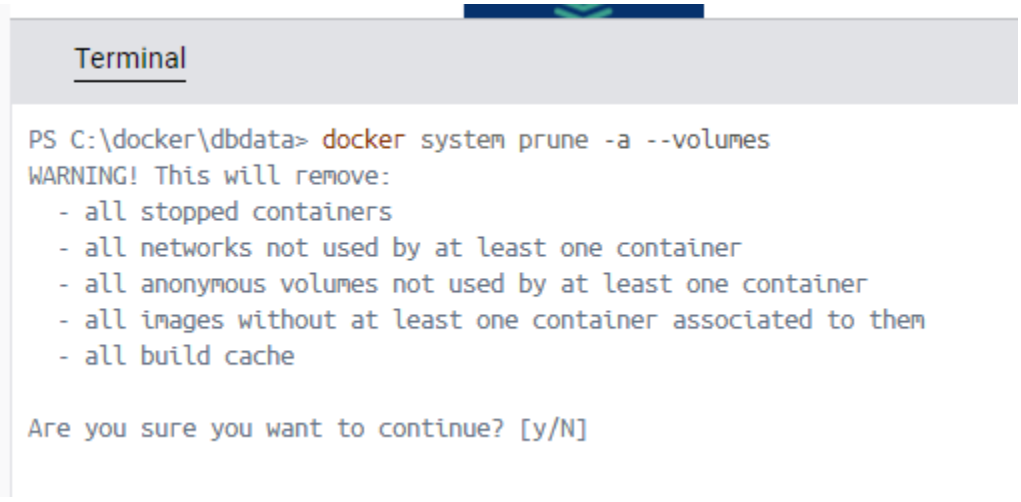
A combinação docker-compose up -d –build, mais habitual para iniciar um ambiente de desenvolvimento depois de fazer alterações no Dockerfile.

Para “deitar abaixo” os serviços: docker-compose down

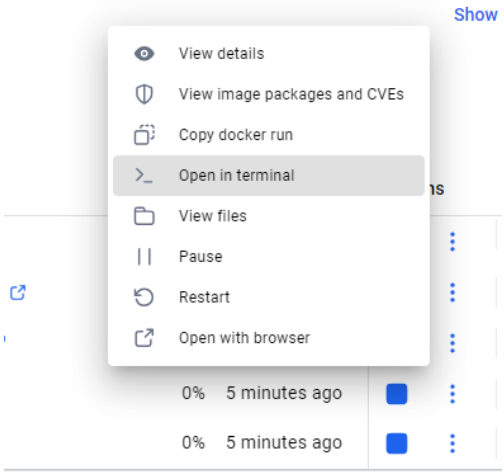


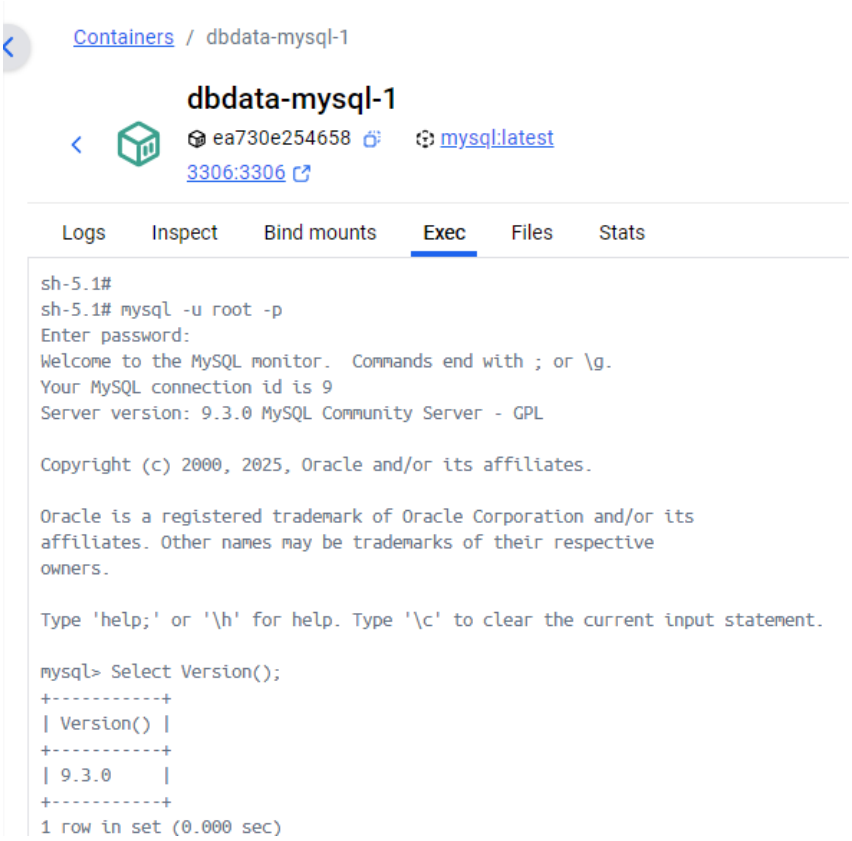
Para remover todos os containers e imagens (para, por exemplo, voltar a criar de novo), depois de assegurar que os containers estão parados:

docker system prune -a --volumes

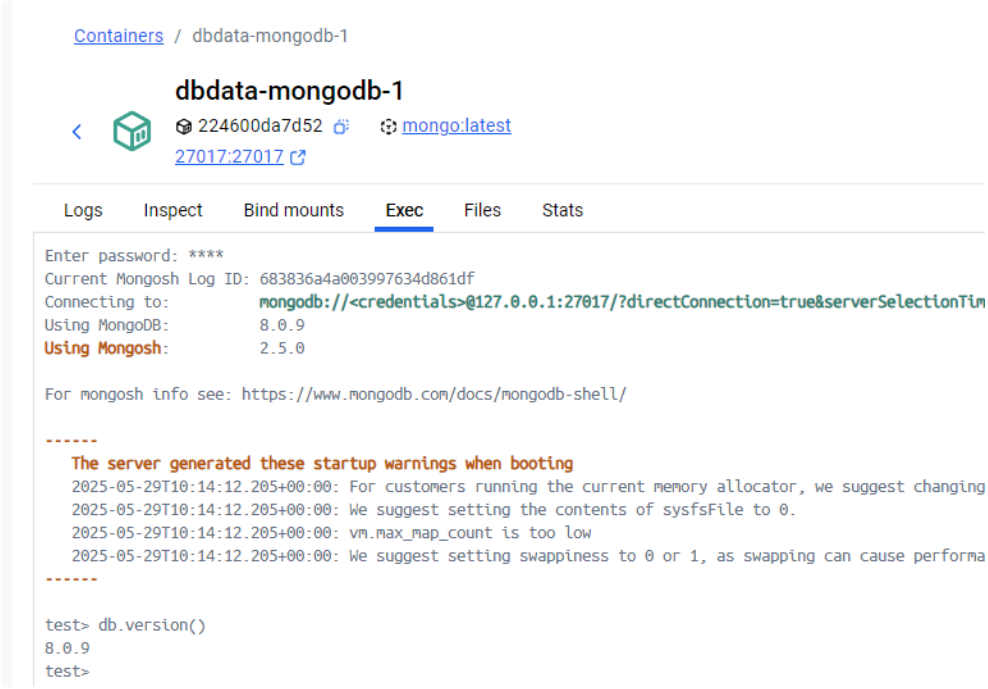


Para abrir o terminal Mysql selecionar na linha do container Mysql-1 os “três pontos” e selecionar Open in terminal.





Para abrir o terminal MongoDB selecionar na linha do container MongoDB-1 os “três pontos” e selecionar Open in terminal. Ou, alternativa, no terminal: docker-compose exec mongodb mongosh --host 127.0.0.1:27017 (ou docker-compose exec mongodb mongosh)



Caso o docker fique corrupto e tenha de ser recriado e queremos recuperar as bases de dados

Assumindo que existe cópia das pastas locais mongo_data e mysql_data (c:\docker\copia)

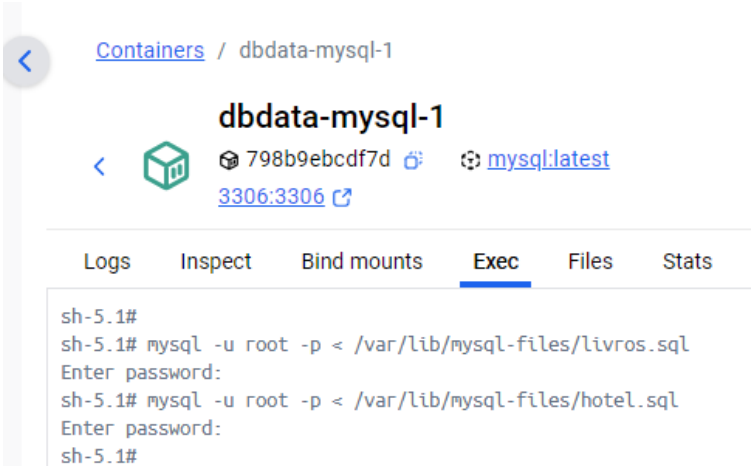
1. docker-compose down -v --rmi all
2. docker system prune -a --volumes -f # -f
   (cuidado, o -v apaga os volumes, inclusive os utilizadores criados)
3. apagar conteúdo de pasta c:\docker\dbdata\mysql_data
4. copiar conteúdo de pasta c:\docker\copia\ dbdata\mysql_data para pasta c:\docker\dbdata\mysql_data
5. a mesma coisa para mongo (3 e 4)
6. docker-compose up -d --build

Mysql

Importar Base de Dados Livros e Base de Dados Hotel (SCRIPT)

```
mysql -u root -p < /var/lib/mysql-files/livros.sql
```

```
mysql -u root -p < /var/lib/mysql-files/hotel.sql
```



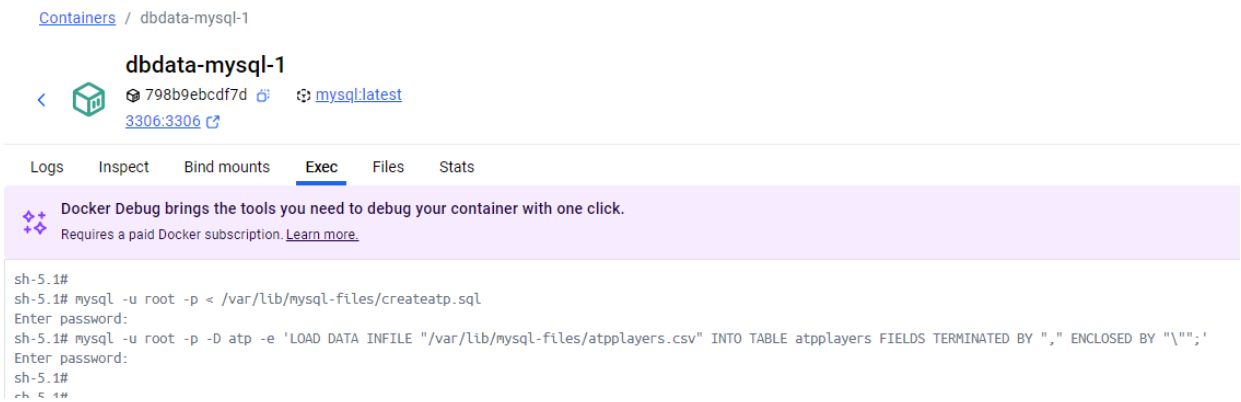
Executar Comandos SQL

```
sh-5.1# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 146
Server version: 9.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use hotel;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from hotel;
+-------------+-------------+
| Sigla_Hotel | Designacao |
+-------------+-------------+
| AL          | Alfa       |
| BH          | Baia       |
| LS          | Lisboa     |
```

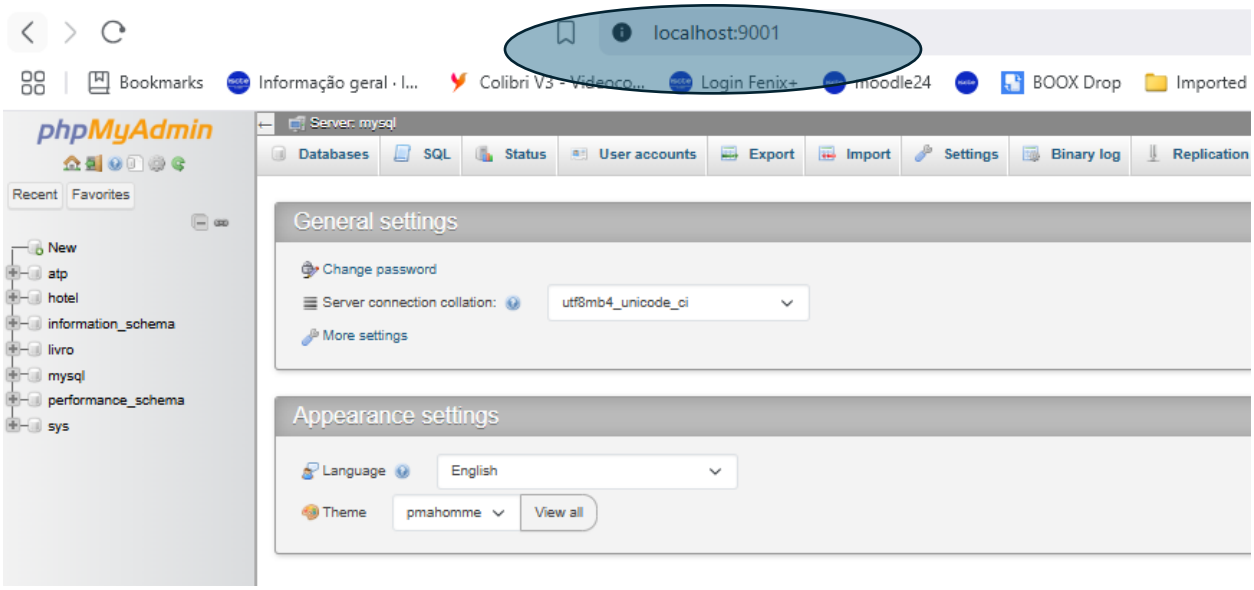
Importar Base de Dados ATP (CSV)

1) Importar script que cria a base de dados e a tabela
   `mysql -u root -p < /var/lib/mysql-files/createatp.sql`
2) Importar o ficheiro com registos csv para a tabela

```
mysql -u root -p -D atp -e 'LOAD DATA INFILE "/var/lib/mysql-
files/atpplayers.csv" INTO TABLE atpplayers FIELDS TERMINATED BY ","
ENCLOSED BY "\"";'
```



PhpMyAdmin



Ligação de Python a Mysql

```
import pymysql
from mysql.connector import Error
import mysql.connector as mariadb

usermysql="root"
passmysql="root"
hostmysql="localhost"
database="hotel"

try:
    connection = mariadb.connect(host=hostmysql, user=usermysql,
passwd=passmysql, db=database,connect_timeout=1000,autocommit=True)
    print("Connected to MySQL DOCKET ATP mp")
except Error as e:
    print("Error while connecting to MySQL Docker DBLivros", e)

sql = "SELECT count(*) as total from hotel;"
try:
    cursor = connection.cursor()
    cursor.execute(sql)
    records = cursor.fetchall()
    for row in records:
        print(row[0])
except:
    print ("Error: unable to SELECT count(*) as total from hotel")
```

Mongo DB

Importar Base de Dados Livros (JSON)

```
mongoimport --host localhost --port 27017 --username root --password
root --authenticationDatabase admin --db bdlivros  --collection livros
--file /tmp/import/livro.json
```

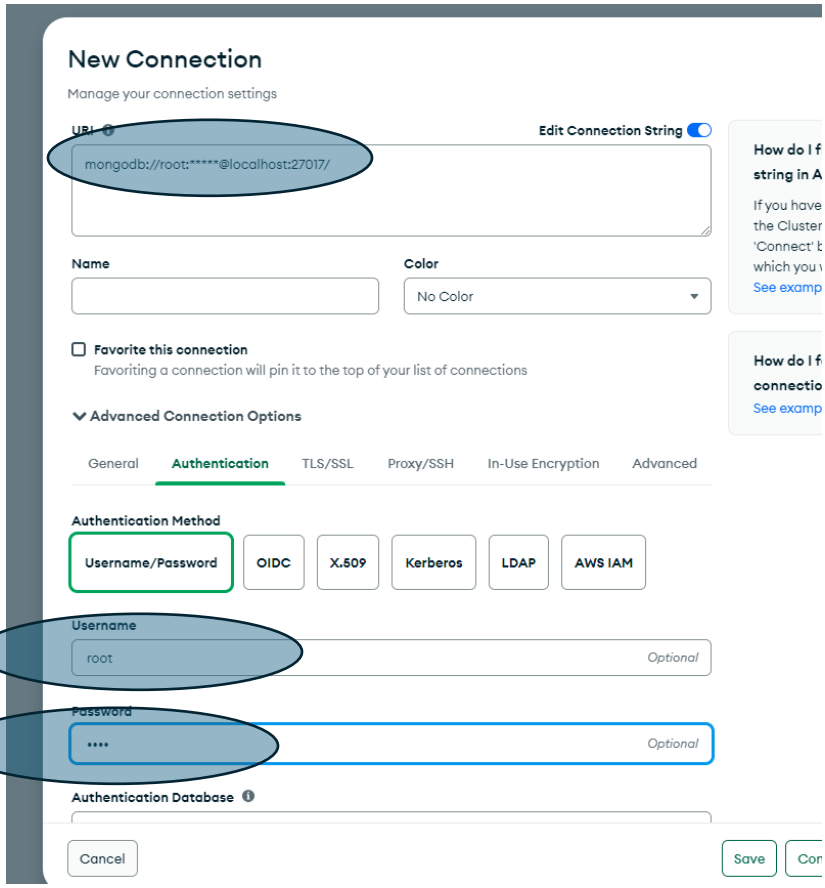
Executar Comandos

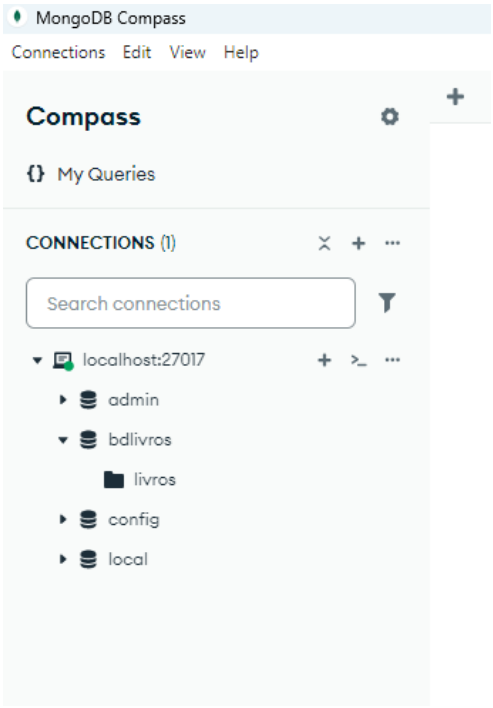
```
#
# mongosh -u root -p
Enter password: ****
Current Mongosh Log ID: 68381a7729d43755e2d861df
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.0
Using MongoDB:          8.0.9
Using Mongosh:          2.5.0

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-05-28T20:58:11.729+00:00: For customers running the current memory allocator, we suggest changing the contents of the following sysfsFile
   2025-05-28T20:58:11.729+00:00: We suggest setting the contents of sysfsFile to 0.
   2025-05-28T20:58:11.729+00:00: vm.max_map_count is too low
   2025-05-28T20:58:11.729+00:00: We suggest setting swappiness to 0 or 1, as swapping can cause performance problems.
------

test> use bdlivros
switched to db bdlivros
bdlivros> db.livros.find()
[
  {
    _id: ObjectId('68381988ade90a671628b3d7'),
    IDLIVRO: 1,
```

O servidor mongod é iniciado automaticamente quando o container é *iniciado*.

Compass

O Compass é executado no Windows





Importar Base de Dados Atp (CSV)

```
mongoimport --host localhost --port 27017 --username root --
password root --authenticationDatabase admin --db atp --collection
atpplayers --type csv --headerline --file
/tmp/import/atpplayers.csv
```

```
#
# mongoimport  --host localhost --port 27017  --username root  --password root  --authenticationDatabase admin
2025-05-29T08:54:36.505+0000    connected to: mongodb://localhost:27017/
2025-05-29T08:54:39.504+0000    [###.....................] atp.atpplayers       30.9MB/219MB (14.1%)
2025-05-29T08:54:42.504+0000    [#####....................] atp.atpplayers       53.4MB/219MB (24.4%)
2025-05-29T08:54:45.504+0000    [########.................] atp.atpplayers       73.5MB/219MB (33.6%)
2025-05-29T08:54:48.504+0000    [###########..............] atp.atpplayers       95.7MB/219MB (43.7%)
2025-05-29T08:54:51.504+0000    [##############...........] atp.atpplayers       121MB/219MB (55.1%)
2025-05-29T08:54:54.504+0000    [#################........] atp.atpplayers       148MB/219MB (67.4%)
2025-05-29T08:54:57.504+0000    [####################.....] atp.atpplayers       174MB/219MB (79.4%)
2025-05-29T08:55:00.504+0000    [#######################..] atp.atpplayers       202MB/219MB (92.2%)
2025-05-29T08:55:02.458+0000    [########################] atp.atpplayers       219MB/219MB (100.0%)
2025-05-29T08:55:02.460+0000    1308835 document(s) imported successfully. 0 document(s) failed to import.
#
```

Ligação de Python a Mongo

```
from pymongo import MongoClient

from pymongo.errors import ConnectionFailure, OperationFailure

db_name = 'atp'
collection_name = 'atpplayers'
client = MongoClient('localhost', 27017, username='root',
password='root', authSource='admin')
#client =
MongoClient('localhost:27017,localhost:27018,localhost:27019',
replicaSet='ReplicaLivros', username='root', password='root',
authSource='admin')
try:
    db = client[db_name]
    collection = db[collection_name]
    num_documentos = collection.count_documents({})
    print(f"Número de documentos: {num_documentos}")

except Exception as e:
    print(f"Ocorreu um erro inesperado: {e}")
finally:
    client.close()
    print("Conexão ao MongoDB fechada.")
```

Réplicas

Para ter vários servidores tenho de ter docker-compose.yml para o réplica set (ver exemplo na pasta)- Depois de construir os containers, abro um terminal numa réplica mongo e

```
use admin;
var config = {
      _id : "Replica",
      members : [
      {_id : 0, host : "mongodb1:27017"},
      {_id : 1, host : "mongodb2:27017"},
      {_id : 2, host : "mongodb3:27017"} ] };
rs.initiate(config);
```

Uma configuração pode ter um mongo standalone e répicas, tuno no mesmo ficheiro.

```
version: '3.9'
services:
...
  mongodb_standalone:
    image: mongo:latest
    container_name: mongodb-standalone
    restart: always
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: root
    volumes:
      - ./mongo-data-standalone:/data/db
      - ./mongostandalone_files:/tmp/import
    networks:
      - app_network


  mongodb1:
    image: mongo:latest
    container_name: mongodb1
    ports:
      - "27018:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: root
    volumes:
      - mongo1_data:/data/db
      - ./mongoreplica_files:/tmp/import
      - ./mongo-keyfile:/data/db/mongo-keyfile
    command: mongod --replSet Replica --bind_ip_all --keyFile
/data/db/mongo-keyfile
    networks:
```

```
      - app_network

  mongodb2:
    image: mongo:latest
    container_name: mongodb2
    ports:
      - "27019:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: root
    volumes:
      - mongo2_data:/data/db
      - ./mongoreplica_files:/tmp/import
      - ./mongo-keyfile:/data/db/mongo-keyfile
    command: mongod --replSet Replica --bind_ip_all --keyFile
/data/db/mongo-keyfile
    networks:
      - app_network

  mongodb3:
    image: mongo:latest
    container_name: mongodb3
    ports:
      - "27020:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: root
    volumes:
      - mongo3_data:/data/db
      - ./mongoreplica_files:/tmp/import
      - ./mongo-keyfile:/data/db/mongo-keyfile
    command: mongod --replSet Replica --bind_ip_all --keyFile
/data/db/mongo-keyfile
    networks:
      - app_network

volumes:
  mongo1_data:
  mongo2_data:
  mongo3_data:

networks:
  app_network:
    driver: bridge
```