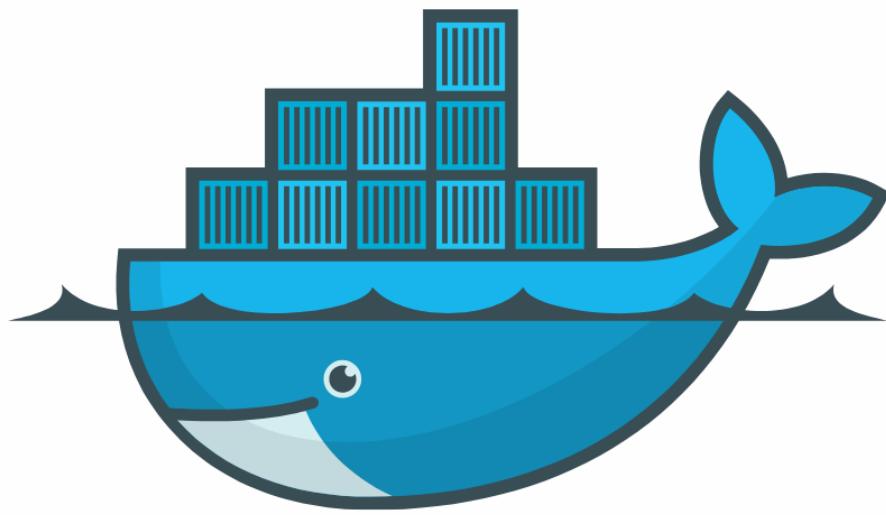


VOLÚMENES: PERSISTIENDO DATOS EN MONGODB



Ivana Sánchez Pérez

Introducción

MongoDB es una base de datos NoSQL orientada a documentos. Su imagen oficial en Docker permite ejecutarlo fácilmente en contenedores.

Por defecto:

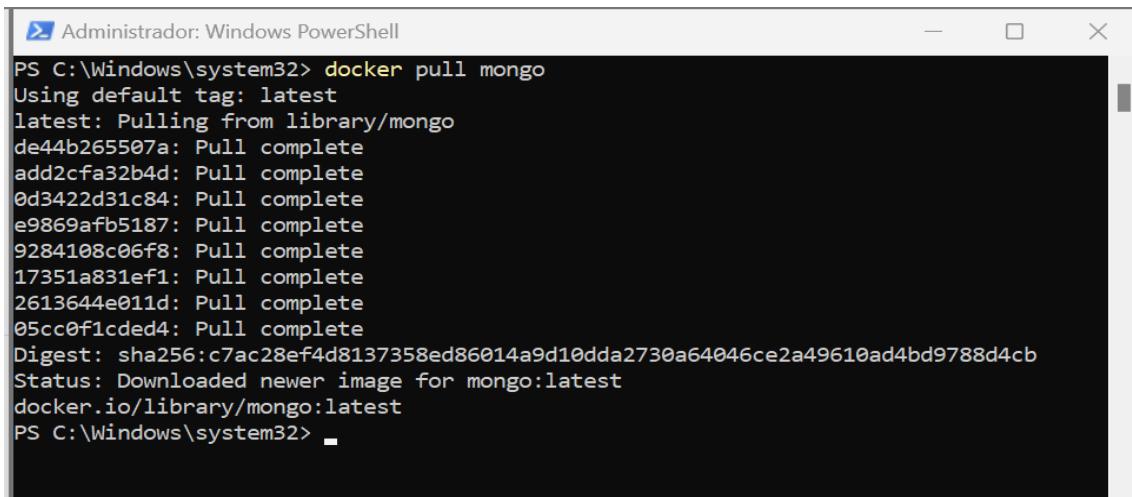
- MongoDB almacena sus datos en **/data/db** dentro del contenedor.
- MongoDB escucha en el puerto 27017

OBJETIVO

Ejecutar un contenedor de MongoDB y montar un volumen persistente en la ruta **/opt/actividades/docker/volumenes**, para evitar la pérdida de datos si el contenedor se elimina.

Descargar la imagen oficial de MongoDB

Ejecutamos el comando **docker pull mongo** para descargar la última versión de la imagen MongoDB.

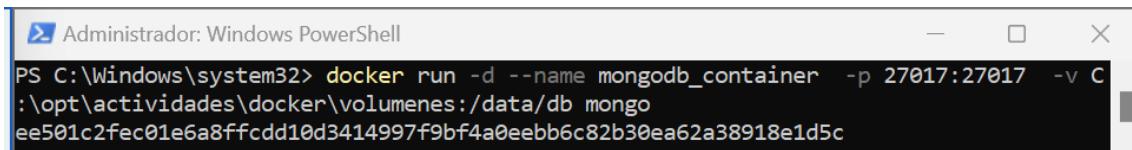


```
PS C:\Windows\system32> docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
de44b265507a: Pull complete
add2cfa32b4d: Pull complete
0d3422d31c84: Pull complete
e9869afb5187: Pull complete
9284108c06f8: Pull complete
17351a831ef1: Pull complete
2613644e011d: Pull complete
05cc0f1cded4: Pull complete
Digest: sha256:c7ac28ef4d8137358ed86014a9d10dda2730a64046ce2a49610ad4bd9788d4cb
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest
PS C:\Windows\system32>
```

Crear un volumen para persistencia de datos

Vamos a ejecutar un contenedor con MongoDB y asignar un volumen a la ruta **/opt/actividades/docker/volumenes** para que los datos no se pierdan.

docker run -d --name mongodb_container -p 27017:27017 -v C:\opt\actividades\docker\volumenes:/data/db mongo



```
PS C:\Windows\system32> docker run -d --name mongodb_container -p 27017:27017 -v C:\opt\actividades\docker\volumenes:/data/db mongo
ee501c2fec01e6a8ffccdd10d3414997f9bf4a0eebb6c82b30ea62a38918e1d5c
PS C:\Windows\system32>
```

Explicación del comando:

- -d → Ejecuta el contenedor en segundo plano.
- --name mongodb_container → Nombre del contenedor.
- -p 27017:27017 → Expone el puerto 27017 en el host.
- -v C:\opt\actividades\docker\volumenes:/data/db → Monta un volumen local en la ruta /data/db, donde MongoDB almacena sus datos.
- mongo → Usa la imagen oficial de MongoDB.

Verificar que MongoDB está corriendo

Ejecutamos el comando **`docker ps`** y para ver los registros y confirmar que se inició correctamente ejecutaremos el comando **`docker logs`**
`mongodb_container`

```
PS C:\Windows\system32> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED             STATUS
              PORTS     NAMES
ee501c2fec01   mongo      "docker-entrypoint.s..."  About a minute ago   Up About a minute
                0.0.0.0:27017->27017/tcp   mongodb_container
75cb76da0b26   nginx      "/docker-entrypoint...."  34 minutes ago    Up 34 minutes
                0.0.0.0:8080->80/tcp   nginx_logs
PS C:\Windows\system32> docker logs mongodb_container
{"t": {"$date": "2025-01-31T11:55:37.817+00:00"}, "s": "I", "c": "CONTROL", "id": 23285, "ctx": "main", "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t": {"$date": "2025-01-31T11:55:37.821+00:00"}, "s": "I", "c": "CONTROL", "id": 5945603, "ctx": "main", "msg": "Multi threading initialized"}
{"t": {"$date": "2025-01-31T11:55:37.840+00:00"}, "s": "I", "c": "NETWORK", "id": 4648601, "ctx": "main", "msg": "Implicit TCP FastOpen unavailable. If TCP FastOpen is required, set at least one of the related parameters", "attr": {"relatedParameters": ["tcpFastOpenServer", "tcpFastOpenClient", "tcpFastOpenQueueSize"]}}
{"t": {"$date": "2025-01-31T11:55:37.847+00:00"}, "s": "I", "c": "NETWORK", "id": 4915701, "ctx": "main", "msg": "Initialized wire specification", "attr": {"spec": {"incomingExternalClient": {"minWireVersion": 0, "maxWireVersion": 25}, "incomingInternalClient": {"minWireVersion": 0, "maxWireVersion": 25}, "outgoing": {"minWireVersion": 6, "maxWireVersion": 25}, "isInternalClient": true}}}
{"t": {"$date": "2025-01-31T11:55:37.849+00:00"}, "s": "I", "c": "TENANT_M", "id": 7091600, "ctx": "main", "msg": "Starting TenantMigrationAccessBlockerRegistry"}
{"t": {"$date": "2025-01-31T11:55:37.849+00:00"}, "s": "I", "c": "CONTROL", "id": 4615611, "ctx": "initandlisten", "msg": "MongoDB starting", "attr": {"pid": 1, "port": 27017, "dbPath": "/data/db", "architecture": "64-bit", "host": "ee501c2fec01"}}
{"t": {"$date": "2025-01-31T11:55:37.849+00:00"}, "s": "I", "c": "CONTROL", "id": 23403, "ctx": "initandlisten", "msg": "Build Info", "attr": {"buildInfo": {"version": "8.0.4", "gitVersion": "bc35ab4305d9920d9d0491c1c9ef9b72383d31f9", "openSSLVersion": "OpenSSL 3.0.13 30 Jan 2024", "modules": [], "allocator": "tcmalloc-google", "environment": {"distmod": "ubuntu2404", "distarch": "x86_64", "target_arch": "x86_64"}}}
{"t": {"$date": "2025-01-31T11:55:37.849+00:00"}, "s": "I", "c": "CONTROL", "id": 51765, "ctx": "initandlisten", "msg": "Operating System", "attr": {"os": {"name": "Ubuntu", "version": "24.04"}}}
{"t": {"$date": "2025-01-31T11:55:37.849+00:00"}, "s": "I", "c": "CONTROL", "id": 21951, "ctx": "initandlisten", "msg": "Options set by command line", "attr": {"options": {"net": {"bindIp": "*"}}}}
{"t": {"$date": "2025-01-31T11:55:37.860+00:00"}, "s": "I", "c": "STORAGE", "id": 22315, "ctx": "initandlisten", "msg": "Opening WiredTiger", "attr": {"config": "create,cache_s"}}
```

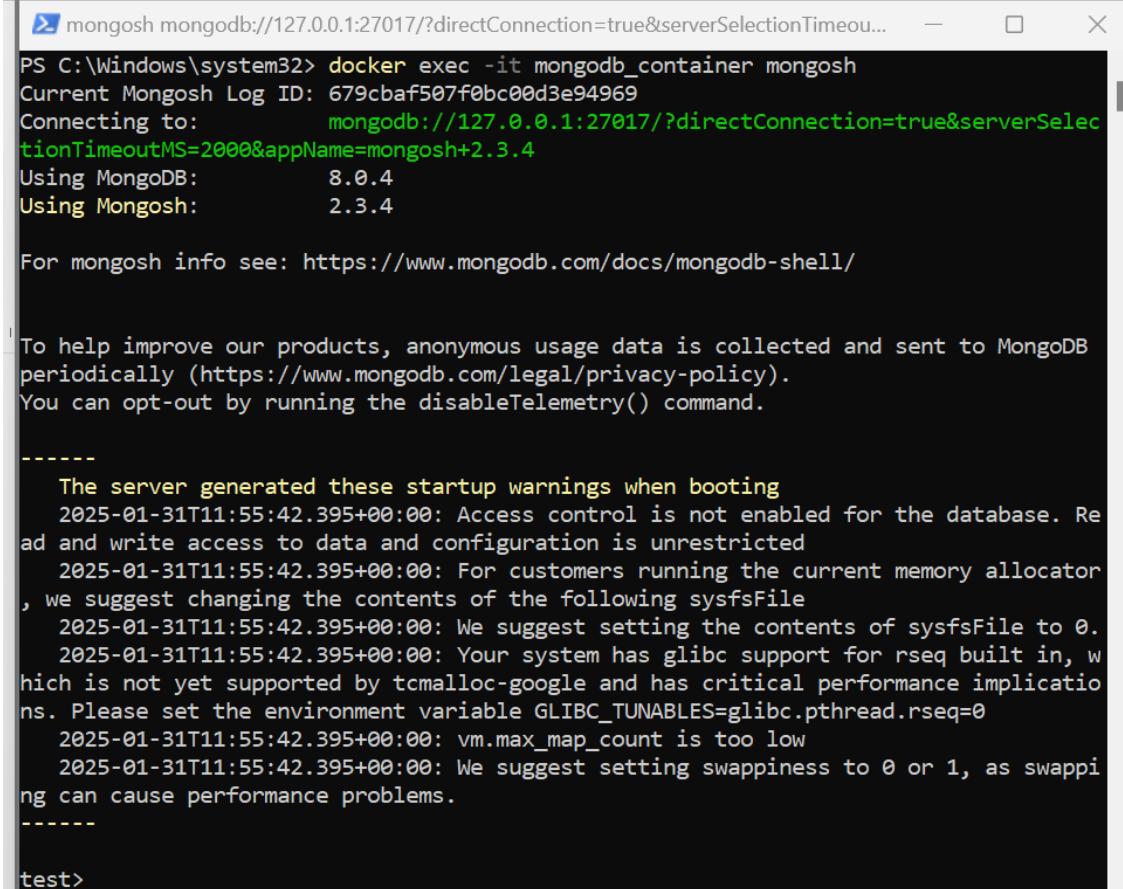
Crear datos en MongoDB

Abrimos la consola con **docker exec -it mongodb_container mongosh** y creamos una base de datos con unos datos para poder realizar la tarea.

```
use miBaseDeDatos
```

```
db.usuarios.insertOne({ nombre: "Ivana", edad: 30 })
```

```
db.usuarios.find()
```



The terminal window shows the following sequence of commands and output:

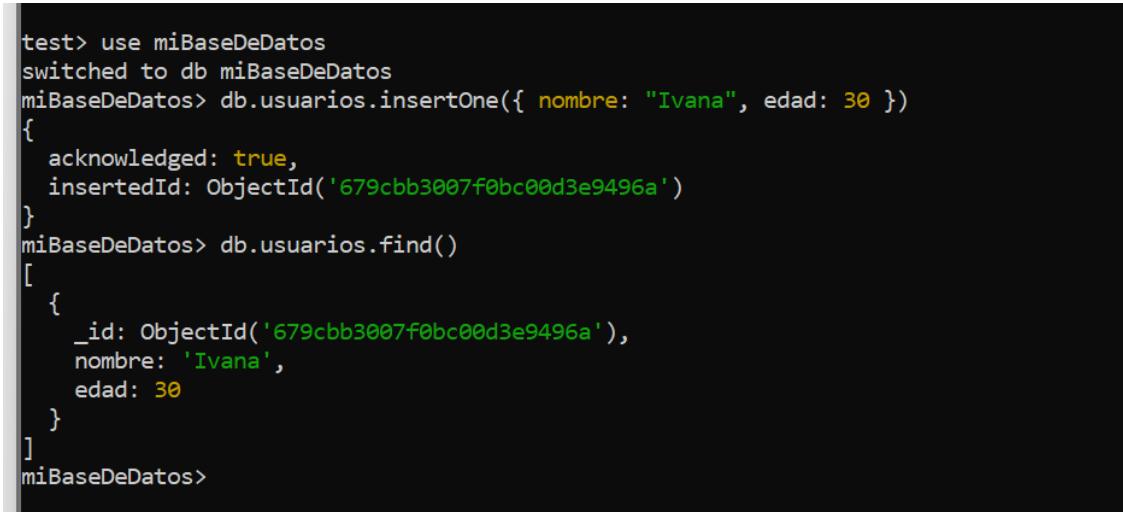
```
PS C:\Windows\system32> docker exec -it mongodb_container mongosh
Current Mongosh Log ID: 679cbb3007f0bc00d3e94969
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.4
Using MongoDB:     8.0.4
Using Mongosh:     2.3.4

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

To help improve our products, anonymous usage data is collected and sent to MongoDB
periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2025-01-31T11:55:42.395+00:00: Access control is not enabled for the database. Re
ad and write access to data and configuration is unrestricted
2025-01-31T11:55:42.395+00:00: For customers running the current memory allocator
, we suggest changing the contents of the following sysfsFile
2025-01-31T11:55:42.395+00:00: We suggest setting the contents of sysfsFile to 0.
2025-01-31T11:55:42.395+00:00: Your system has glibc support for rseq built in, w
hich is not yet supported by tcmalloc-google and has critical performance implicatio
ns. Please set the environment variable GLIBC_TUNABLES=glibc.pthread.rseq=0
2025-01-31T11:55:42.395+00:00: vm.max_map_count is too low
2025-01-31T11:55:42.395+00:00: We suggest setting swappiness to 0 or 1, as swappi
ng can cause performance problems.
-----

test>
```

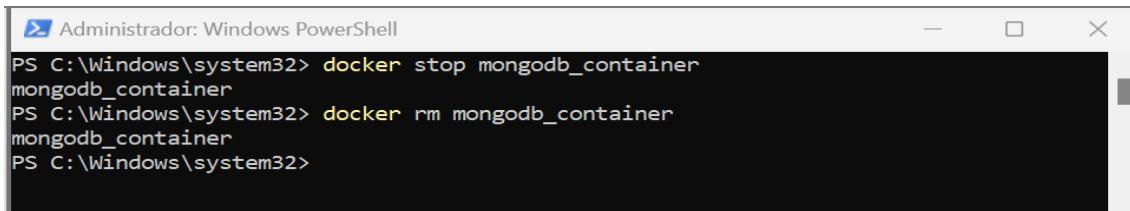



```
test> use miBaseDeDatos
switched to db miBaseDeDatos
miBaseDeDatos> db.usuarios.insertOne({ nombre: "Ivana", edad: 30 })
{
  acknowledged: true,
  insertedId: ObjectId('679cbb3007f0bc00d3e9496a')
}
miBaseDeDatos> db.usuarios.find()
[
  {
    _id: ObjectId('679cbb3007f0bc00d3e9496a'),
    nombre: 'Ivana',
    edad: 30
  }
]
miBaseDeDatos>
```

Comprobar que los datos persisten tras eliminar el contenedor

Eliminamos el contenedor

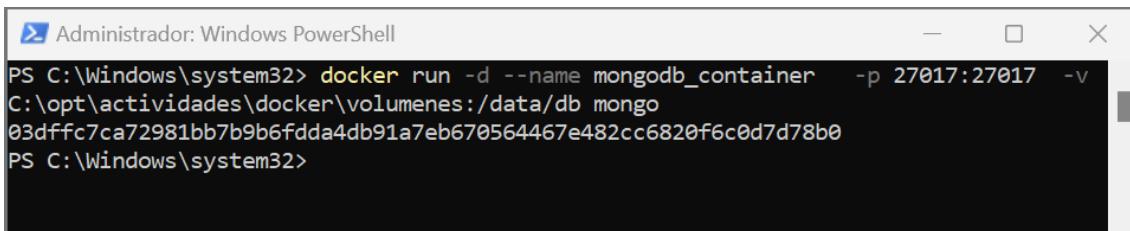
- Salimos de MongoDB con exit
- **docker stop mongodb_container**
- **docker rm mongodb_container**



```
Administrator: Windows PowerShell
PS C:\Windows\system32> docker stop mongodb_container
mongodb_container
PS C:\Windows\system32> docker rm mongodb_container
mongodb_container
PS C:\Windows\system32>
```

Recreamos el contenedor con el mismo volumen

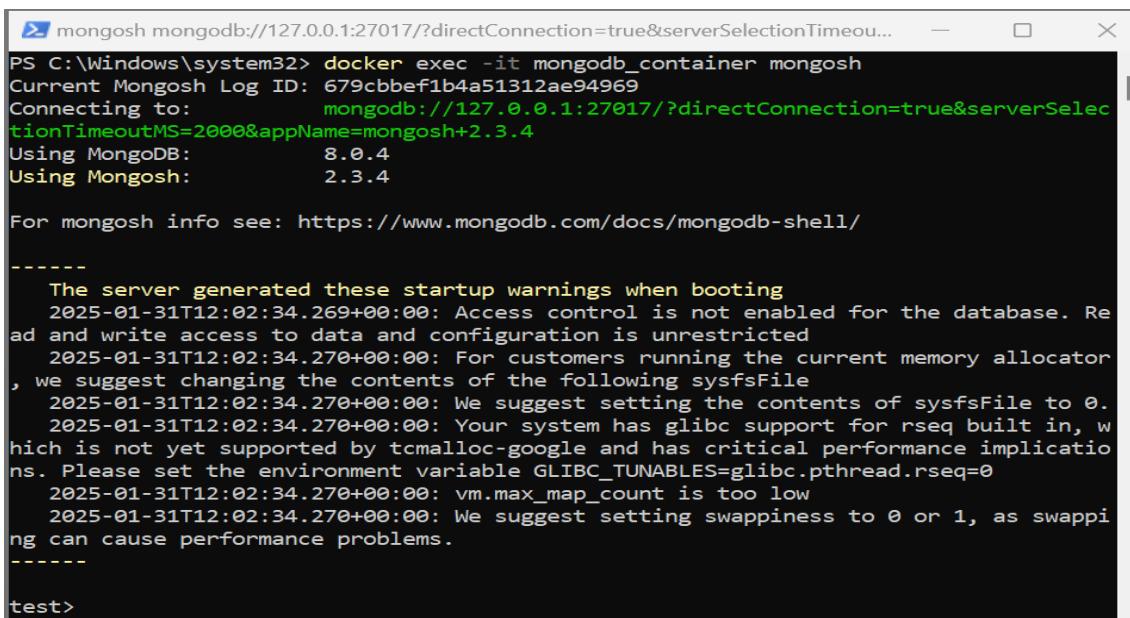
- **docker run -d --name mongodb_container -p 27017:27017 -v C:\opt\actividades\docker\volumenes:/data/db mongo**



```
Administrator: Windows PowerShell
PS C:\Windows\system32> docker run -d --name mongodb_container -p 27017:27017 -v C:\opt\actividades\docker\volumenes:/data/db mongo
03dffcc7ca72981bb7b9b6fdda4db91a7eb670564467e482cc6820f6c0d7d78b0
PS C:\Windows\system32>
```

Verificamos que los datos siguen en la base de datos

- 1- Abrimos de nuevo MongoDB → **docker exec -it mongodb_container mongosh**



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.4
PS C:\Windows\system32> docker exec -it mongodb_container mongosh
Current Mongosh Log ID: 679cbbef1b4a51312ae94969
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.4
Using MongoDB: 8.0.4
Using Mongosh: 2.3.4

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/
-----
The server generated these startup warnings when booting
2025-01-31T12:02:34.269+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2025-01-31T12:02:34.270+00:00: For customers running the current memory allocator, we suggest changing the contents of the following sysfsFile
2025-01-31T12:02:34.270+00:00: We suggest setting the contents of sysfsFile to 0.
2025-01-31T12:02:34.270+00:00: Your system has glibc support for rseq built in, which is not yet supported by tcmalloc-google and has critical performance implications. Please set the environment variable GLIBC_TUNABLES=glibc.pthread.rseq=0
2025-01-31T12:02:34.270+00:00: vm.max_map_count is too low
2025-01-31T12:02:34.270+00:00: We suggest setting swappiness to 0 or 1, as swapping can cause performance problems.
-----
test>
```

2- Consultamos la base de datos→

```
use miBaseDeDatos
db.usuarios.find()
```

```
test> use miBaseDeDatos
switched to db miBaseDeDatos
miBaseDeDatos> db.usuarios.find()
[
  {
    _id: ObjectId('679cbb3007f0bc00d3e9496a'),
    nombre: 'Ivana',
    edad: 30
  }
]
miBaseDeDatos>
```

