

Configurando MongoDB Atlas – mongoose – express – morgan – bcryptjs - jsonwebtoken.

Parte - 2

Para terminar la parte de las rutas, debemos cargar la variable “ruta” en nuestro fichero “aplicacion.js”.

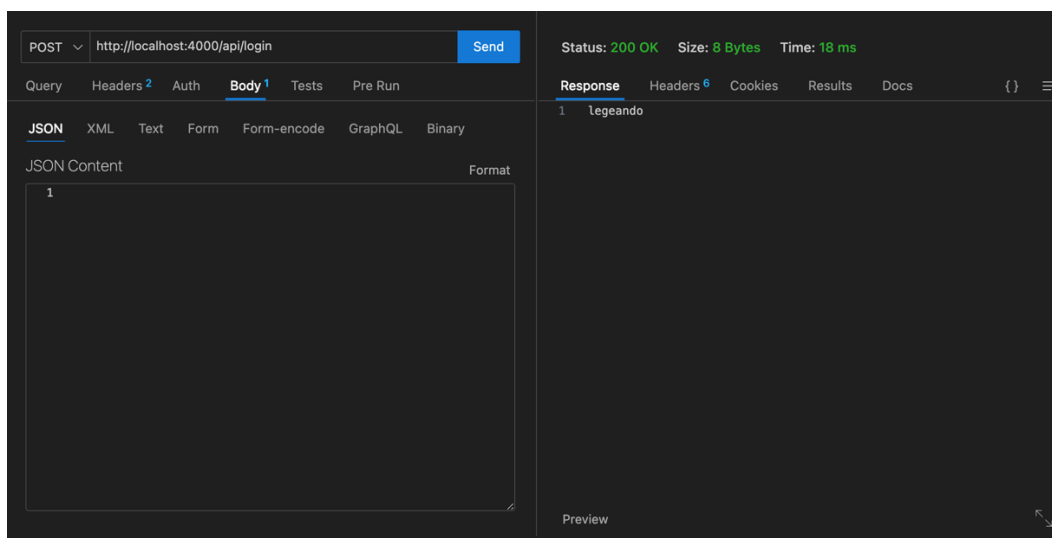
```
import express from 'express'; //importamos la libreria expres.
import morgan from 'morgan'; //Nos sirve para ver las peticiones que se le
hacen al servidor.
import ruta from './rutas/autor.routes.js';
const aplicacion = new express(); //instanciamos.

aplicacion.use(morgan("dev")); //Esta entrada, lo único que nos hace es
mostrarnos un mensaje corto por consola.
aplicacion.use(express.json()); //Para que nuestro pequeño servidor pueda
trabajar con json.
aplicacion.use(ruta);

export default aplicacion; //esto nos va a servir para exportar las variables
a otro .js
```

Para probar que nos funciona correctamente, debemos utilizar un frontend es decir, simular un cliente hasta que desarrollemos el nuestro.

En este caso, vamos a cargar la extensión “**Thunder Client**”, que nos va a simular el frontend.





Cofinanciado por
la Unión Europea



Fondos Europeos



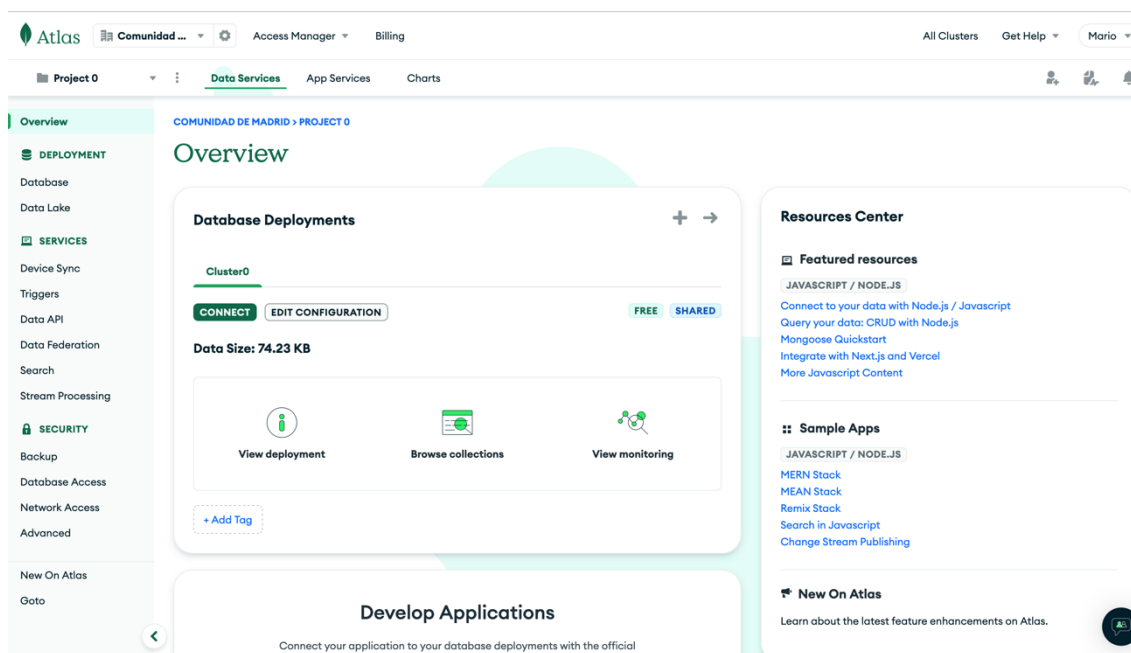
Podemos configurar nuestras rutas desde nuestro fichero “**aplicacion.js**”

```
import express from 'express'; //importamos la libreria expres.
import morgan from 'morgan'; //Nos sirve para ver las peticiones que se le
hacen al servidor.
import ruta from './rutas/autor.routes.js';
const aplicacion = new express(); //instanciamos.

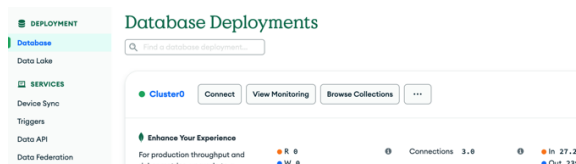
aplicacion.use(morgan("dev")); //Esta entrada, lo único que nos hace es
mostrarnos un mensaje corto por consola.
aplicacion.use(express.json()); //Para que nuestro pequeño servidor pueda
trabajar con json.
aplicacion.use("/api", ruta);

export default aplicacion; //esto nos va a servir para exportar las variables
a otro .js
```

Configurando nuestra BBDD MongoDB



Seleccionamos **Database** y nos encontramos la siguiente pantalla:





Cofinanciado por
la Unión Europea



Fondos Europeos



Si pulsamos sobre “Cluster0”, nos lleva a nuestro espacio de trabajo gratuito.

Si no podemos utilizar en el centro el MongoDB Atlas por temas del proxy, configuramos el mongodb en docker.

Lo primero que hacemos es instalar el Docker para escritorio en nuestra máquina. Desde la página web de Docker, podemos bajarnos el fichero de instalación para Windows, para Mac o para Linux.

En nuestro caso y pensando en las máquinas del centro, vamos a ir al siguiente link y hacemos la instalación:

<https://docs.docker.com/engine/install/ubuntu/>

ANTES DE INSTALAR, COMPROBAR QUE NO TENEMOS INSTALADO EN LAS MÁQUINAS YA EL DOCKER UTILIZANDO EL COMANDO “docker --version”. SI YA LO TENEMOS INSTALADO, LOS SIGUIENTES PASOS NO LOS REALIZAREMOS.

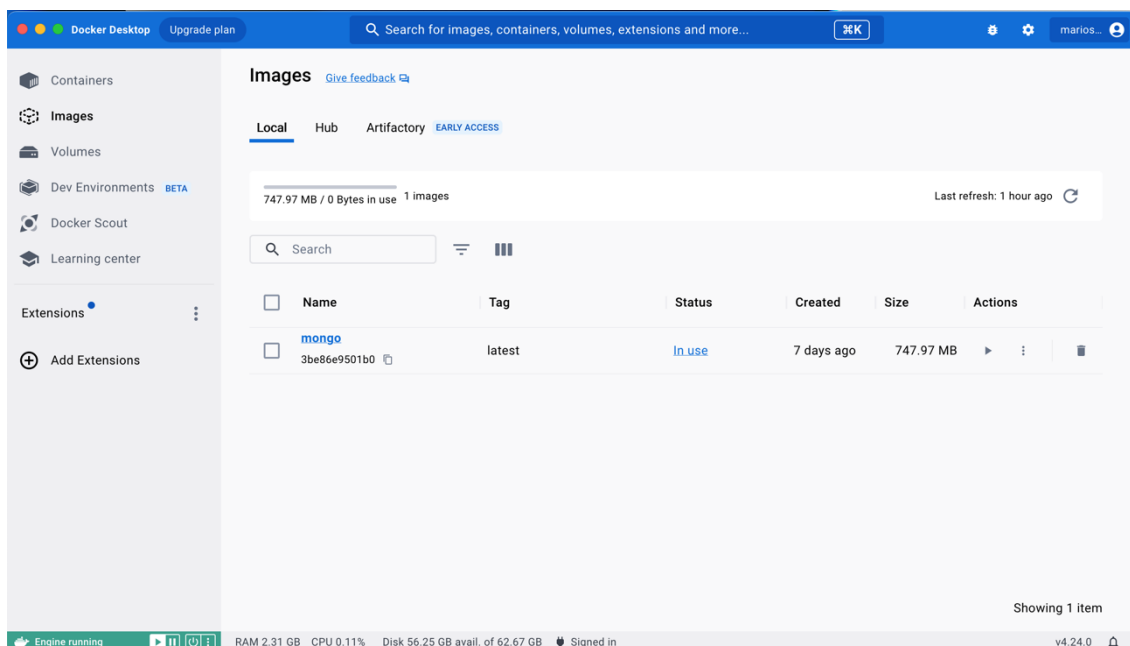
Esto nos crea una interface gráfica de Docker aunque nosotros lo vamos a utilizar desde el terminal.



Cofinanciado por
la Unión Europea

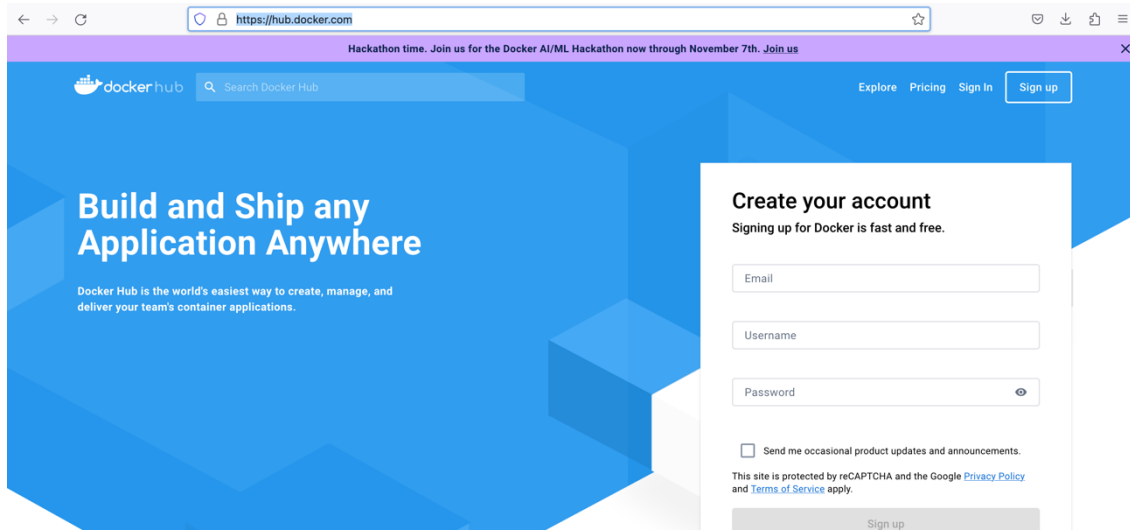


Fondos Europeos



Para hacer un pull de mongo desde Docker, vamos a ir a mongo hub. Desde aquí podremos encontrar las distintas imágenes que nos queremos descargar. En este caso, vamos a buscar mongo:

<https://hub.docker.com/>



No nos hace falta registrarlo, lo único que debemos hacer en esta página es buscar la imagen con la que queremos trabajar.

En este caso, vamos a buscar “mongo”:



Cofinanciado por
la Unión Europea



Fondos Europeos



Hackathon time. Join us for the Docker AI/ML Hackathon now through November 7th. Join us

dockerhub

Explore Pricing Sign In Sign up

Filters 1 - 25 of 10.000 results for mongo. Best Match

Products

- ☐ Images
- ☐ Extensions
- ☐ Plugins

Trusted Content

- ☐ Docker Official Image
- ☐ Verified Publisher
- ☐ Sponsored OSS

Operating Systems

- ☐ Linux
- ☐ Windows

Architectures

- ☐ ARM
- ☐ ARM 64
- ☐ IBM POWER

mongo Docker Official Image · 1B+ · 9.9K
Updated 7 days ago
MongoDB document databases provide high availability and easy scalability.
Windows Linux x86-64 ARM 64 IBM Z

mongo-express Docker Official Image · 100M+ · 1.4K
Updated 8 days ago
Web-based MongoDB admin interface, written with Node.js and express
Linux ARM 64 x86-64

mongodb/mongodb-atlas-kubernetes-operator Verified Publisher · 500K+ · 5
By MongoDB · Updated 4 days ago
The MongoDB Atlas Kubernetes Operator - Kubernetes native management of MongoDB A...
Linux unknown arm64 unknown x86-64

Nos hace una búsqueda y nos muestra varias imágenes que podemos descargar.

Pulsamos sobre la imagen que nos queremos descargar, en este caso, es la primera “mongo” y se nos muestra la siguiente pantalla:

Hackathon time. Join us for the Docker AI/ML Hackathon now through November 7th. Join us

dockerhub

Explore Pricing Sign In Sign up

Explore Official Images mongo

mongo Docker Official Image · 1B+ · 9.9K
MongoDB document databases provide high availability and easy scalability.

Overview Tags

Quick reference

- Maintained by:
the Docker Community
- Where to get help:
the Docker Community Slack, Server Fault, Unix & Linux, or Stack Overflow

Supported tags and respective Dockerfile links

Note: the description for this image is longer than the Hub length limit of 25000, so the "Supported tags" list has been

Recent Tags

latest jammy 7.0.2-jammy 7.0.2 7.0-jammy 7.0
7-jammy 7 6.0.10-jammy 6.0.10

About Official Images

Docker Official Images are a curated set of Docker open source and drop-in solution repositories.

Why Official Images?

These images have clear documentation, promote

docker pull mongo

Lo que nos hace falta es el comando que os he seleccionado en la elipse negra en la imagen anterior. Este comando lo vamos a ejecutar desde el terminal de nuestra máquina y esto nos va a descargar esa imagen.

Si ejecutamos el comando “**docker images**”, nos debe de aparecer la imagen preparada para su uso.

```
mariosantos — zsh — 80x24
[mariosantos@MBP-de-Mario ~ % docker images]
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
mongo         latest    3be86e9501b0   7 days ago    748MB
mariosantos@MBP-de-Mario ~ %
```

Para arrancar mongodb en nuestra máquina y poder trabajar con él de forma local, lanzamos el siguiente comando:

docker run mongo

```
mariosantos — com.docker.cli - docker run mongo — 105x24
msg:"Listening on", "attr":{"address":"/tmp/mongodb-27017.sock"}}
{"t":{"$date":"2023-10-10T10:59:49.154+00:00"},"s":"I", "c":"NETWORK", "id":23015, "ctx":"listener", "msg":"Listening on", "attr":{"address":"0.0.0.0"}}
{"t":{"$date":"2023-10-10T10:59:49.154+00:00"},"s":"I", "c":"NETWORK", "id":23016, "ctx":"listener", "msg":"Waiting for connections", "attr":{"port":27017, "ssl":"off"}}
{"t":{"$date":"2023-10-10T10:59:49.168+00:00"},"s":"I", "c":"REPL", "id":7360102, "ctx":"LogicalSessionCacheRefresh", "msg":"Added oplog entry for createIndexes to transaction", "attr":{"namespace":"config.$cmd", "u uid":{"$uid":{"$uid":"846b37b3-d1c3-4c3e-a283-4db43c629ebc"}}, "object":{"create":"system.sessions", "idIn dex":{"v":2, "key":{"_id":1}, "name":"_id"}}}}
{"t":{"$date":"2023-10-10T10:59:49.168+00:00"},"s":"I", "c":"REPL", "id":7360100, "ctx":"LogicalSessionCacheRefresh", "msg":"Added oplog entry for createIndexes to transaction", "attr":{"namespace":"config.$cmd", "u uid":{"$uid":{"$uid":"846b37b3-d1c3-4c3e-a283-4db43c629ebc"}}, "object":{"createIndexes":"system.s sessions", "v":2, "key":{"lastUse":1}, "name":"lsidTTLIndex", "expireAfterSeconds":1800}}
{"t":{"$date":"2023-10-10T10:59:49.174+00:00"},"s":"I", "c":"INDEX", "id":20345, "ctx":"LogicalSessionCacheRefresh", "msg":"Index build: done building", "attr":{"buildUUID":null, "collectionUUID":{"$uid":{"$uid":"846b37b3-d1c3-4c3e-a283-4db43c629ebc"}}, "namespace":"config.system.sessions", "index":"_id", "ident ":"index-5--7888809240508770750", "collectionIdent":"collection-4--7888809240508770750", "commitTimestamp": null}}
{"t":{"$date":"2023-10-10T10:59:49.174+00:00"},"s":"I", "c":"INDEX", "id":20345, "ctx":"LogicalSessionCacheRefresh", "msg":"Index build: done building", "attr":{"buildUUID":null, "collectionUUID":{"$uid":{"$uid":"846b37b3-d1c3-4c3e-a283-4db43c629ebc"}}, "namespace":"config.system.sessions", "index":"lsidTTLIndex", "ident ":"index-6--7888809240508770750", "collectionIdent":"collection-4--7888809240508770750", "commitTimestamp": null}}
```

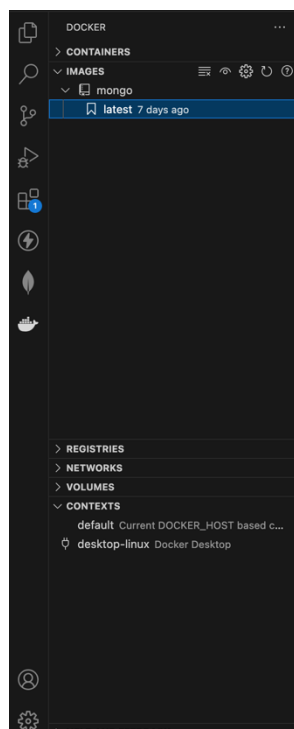
Desde este momento, ya tenemos abierto mongodb escuchando en nuestra máquina y podremos trabajar con él. Si os fijáis en toda la información os dice en que puerto está escuchando mongodb. Por defecto ese puerto es el 27017.

A partir de aquí lo tenemos todo listo para trabajar desde visual code.

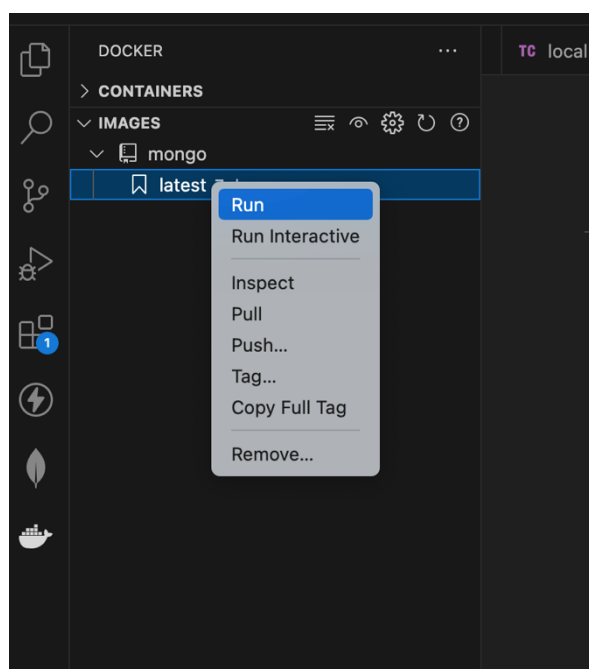
Vamos a instalar la extensión “**Docker**” en visual Code. Esta extensión nos va a permitir gestionar de forma adecuada Docker:

- ver las imágenes cargadas.
- Ver networks

- Etc.



Si todo está correcto, podemos arrancar desde el terminal de code nuestra imagen de mongo o desde la extensión de Docker:



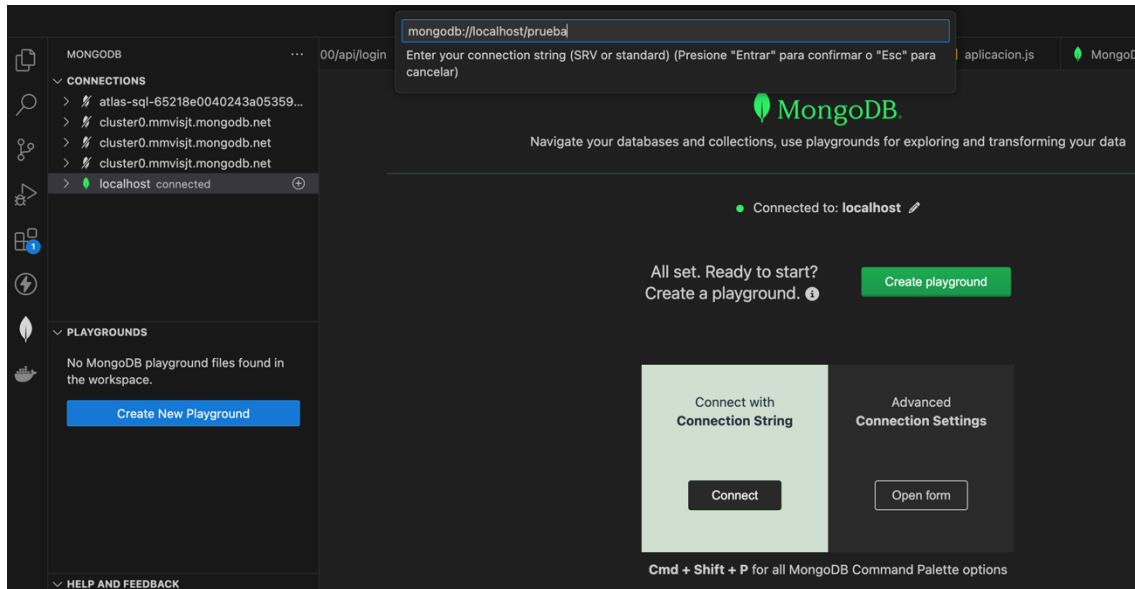
Una vez arrancado, si vamos a la extensión de mongo e intentamos acceder a mongoddb, nos debería de permitir la conexión:



Cofinanciado por
la Unión Europea



Fondos Europeos



Si os fijáis, en el string de conexión a mongo, le paso como último parámetro “prueba”, esta es la bbdd que vamos a crear.