Índice:

Ejercicio 1

1)Instalar docker y docker-compose en una máquina virtual Ubuntu Server

Solución:

Comandos utilizados:

-sudo apt update

-sudo apt upgrade

Paquetes de requisitos previos

-sudo apt-get install curl apt-transport-https ca-certificates software-properties-common

Agregar repositorios

-curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
-sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
-sudo apt update
-apt-cache policy docker-ce

Instalamos Dockers

-sudo apt install docker-ce

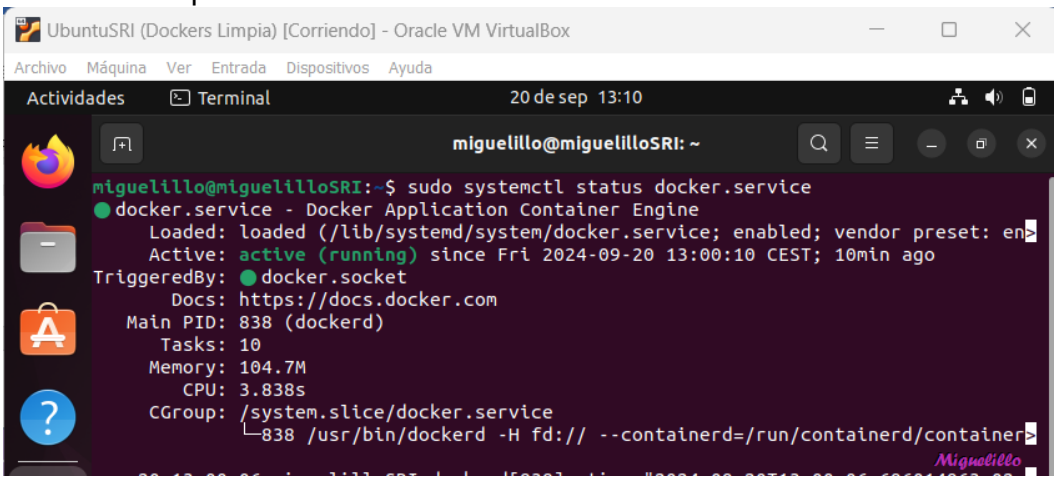
Instalamos docker compose

-mkdir -p ~/.docker/cli-plugins/

-curl -SL https://github.com/docker/compose/releases/download/v2.3.3/docker-compose-linux-x86_64 -o ~/.docker/cli-plugins/docker-compose

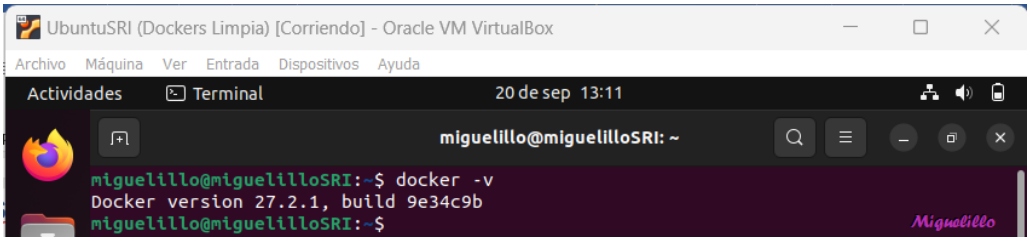
Comprobamos que se instaló

-docker compose version



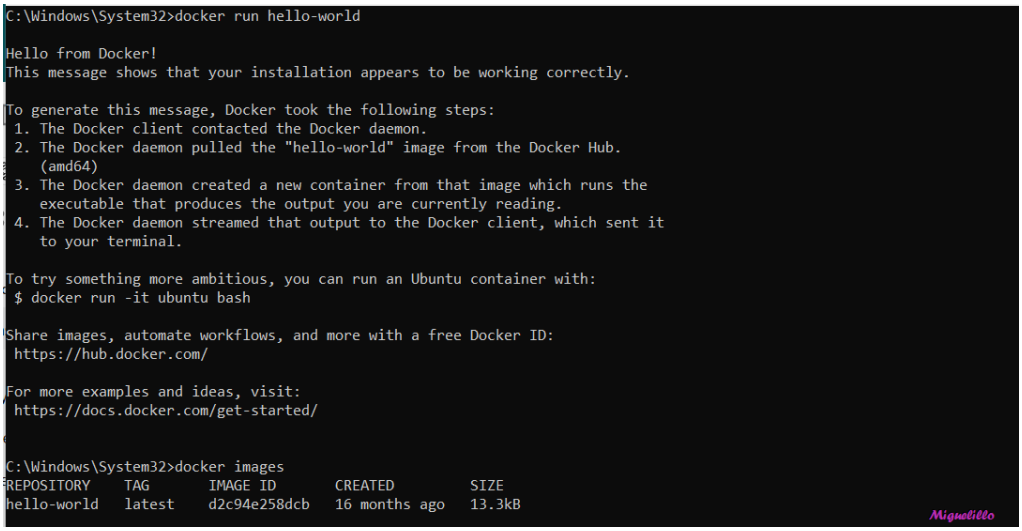
2)Conseguir ejecutar comandos de docker sin necesidad de sudo

Solución:



3)Descargar la imagen Hello-world del repositorio oficial de docker.

Solución:

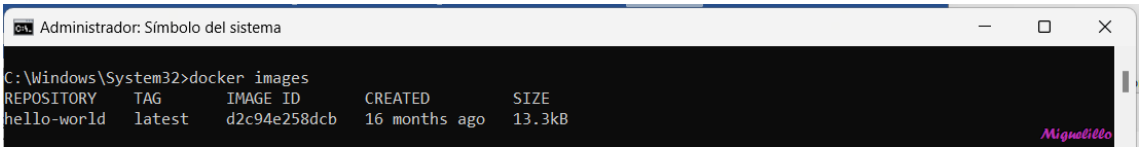


4) Listar las imágenes y contenedores existentes en tu MV, ejecutar la imagen anterior, comprobar su estado y finalmente borrar la imagen de tu MV.

Solución:

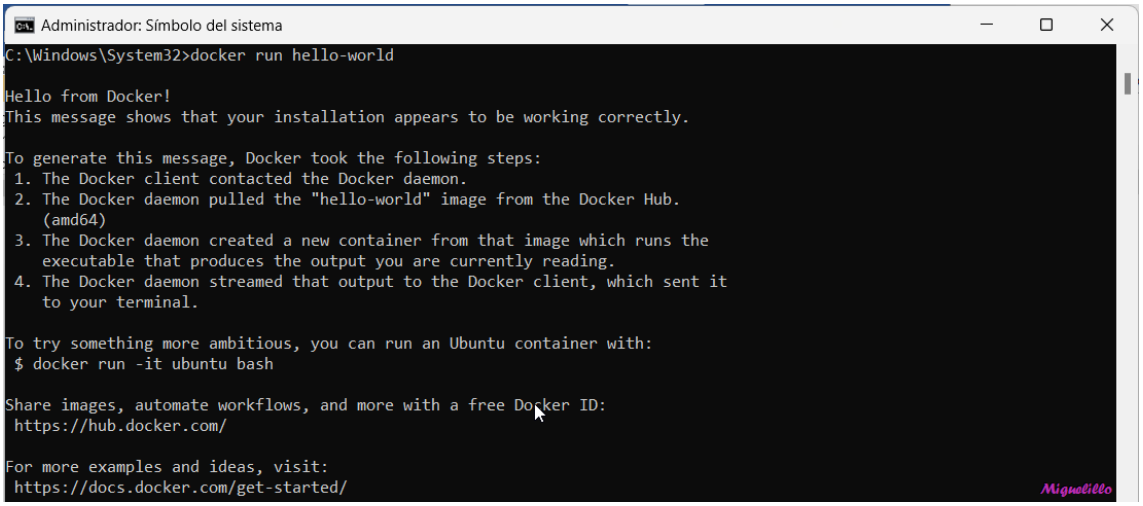
Para listarlas ´

```
docker images
```



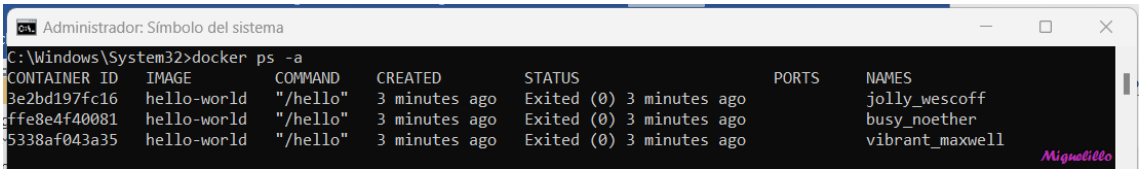
Para ejecutarla:

```
docker run hello-world
```



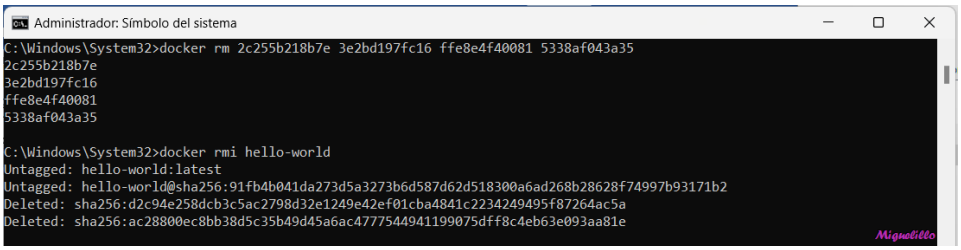
Para comprobar el estado

```
docker ps -a
```



Para eliminarlo, en este caso he tenido que forzar la detención

```
docker rmi hello-world
```

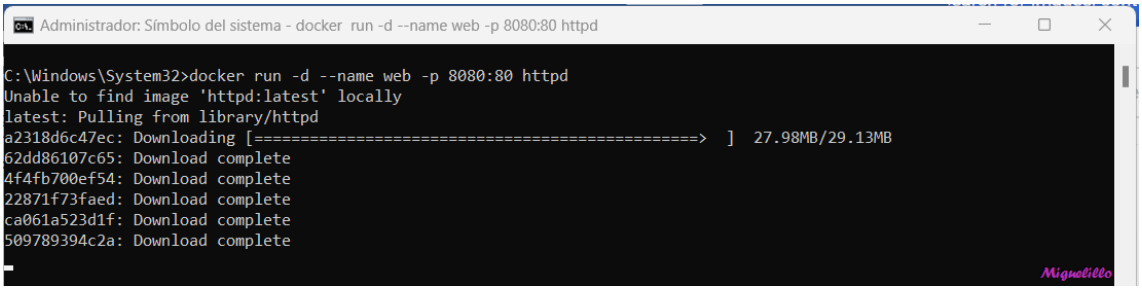


5) Ejecutar el contenedor httpd poniéndole como nombre web y redirigiendo al puerto 8080 del host(tu MV) el puerto 80 del contenedor. Prueba el acceso a la web.

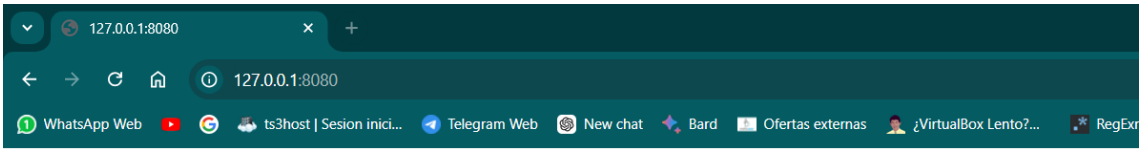
Solución:

Descargamos el contenedor

```
docker run -p 8080:80 –name web httpd
```



Probamos que funcione introduciendo la IP y el puerto

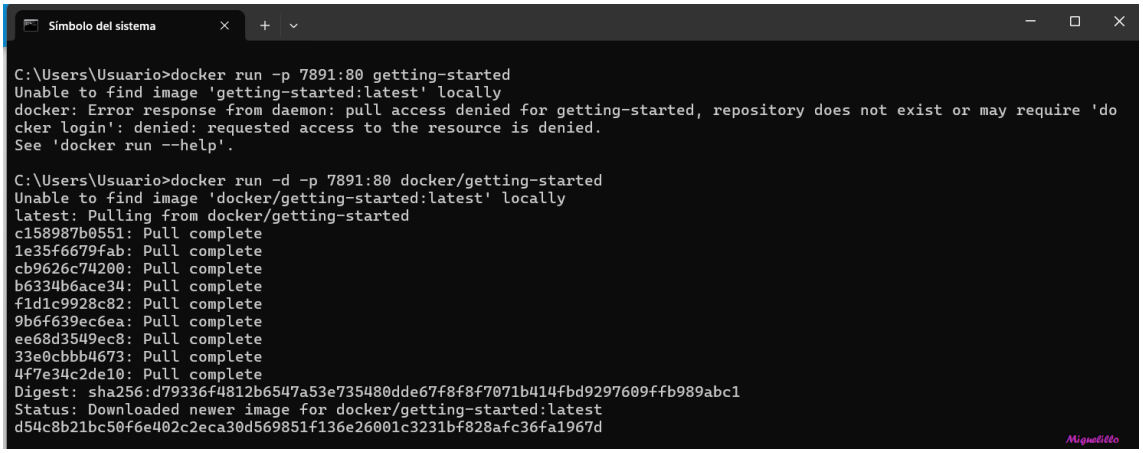


It works!

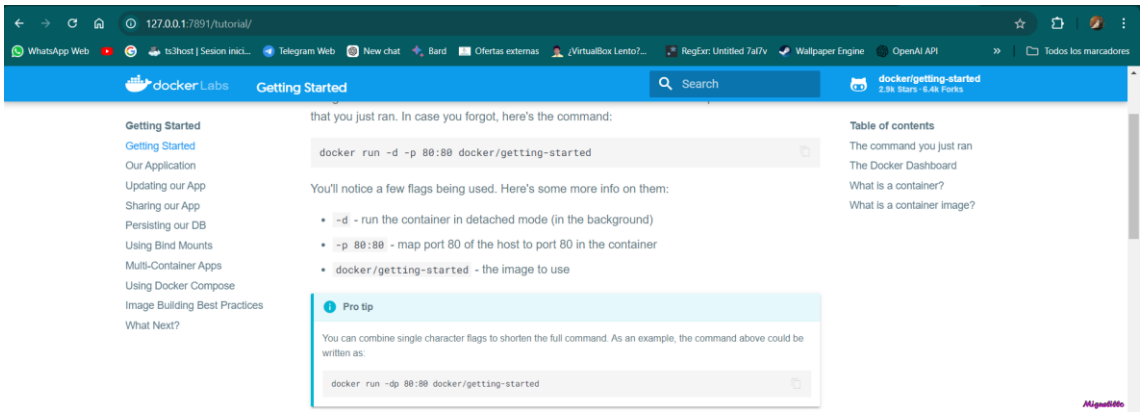
Miguelillo

Instamos el getting started para Windows

```
docker run -d -p 7891:80 docker/getting-started
```



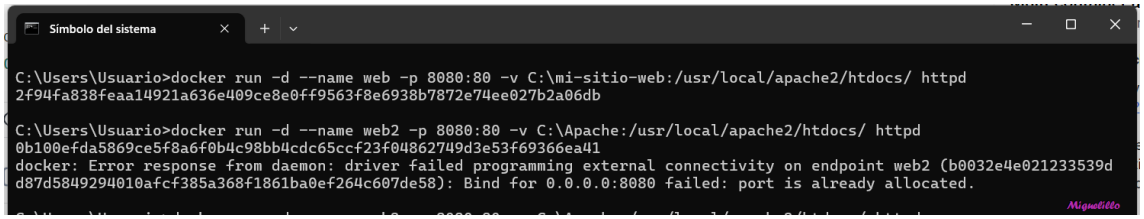
Comprobamos su funcionamiento



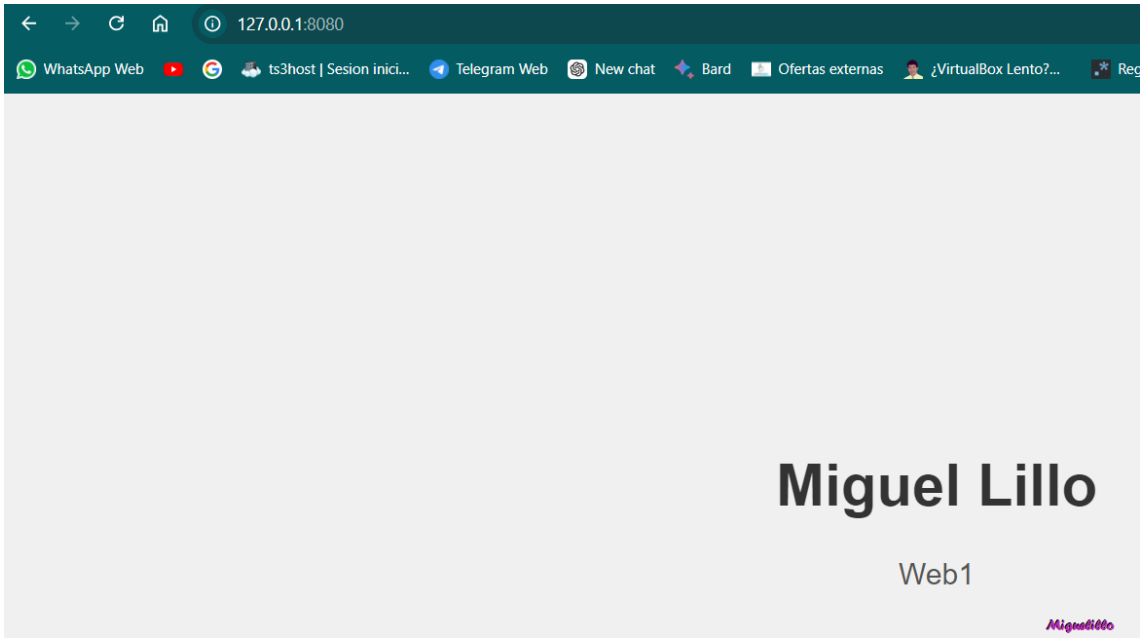
6) Modificar la ejecución anterior para que tu contenido web se encuentre en una carpeta real de tu MV.

Solución:

Instalas httpd y lo creas especificando la con la ruta de los archivos



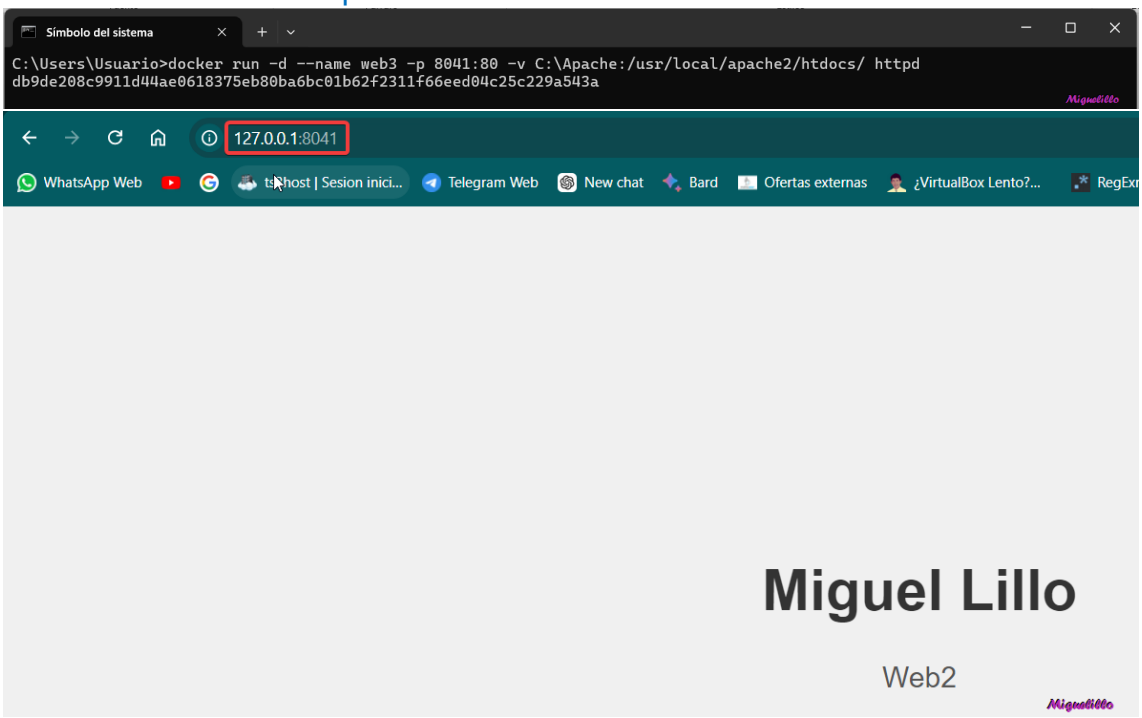
Pruebas que funcione



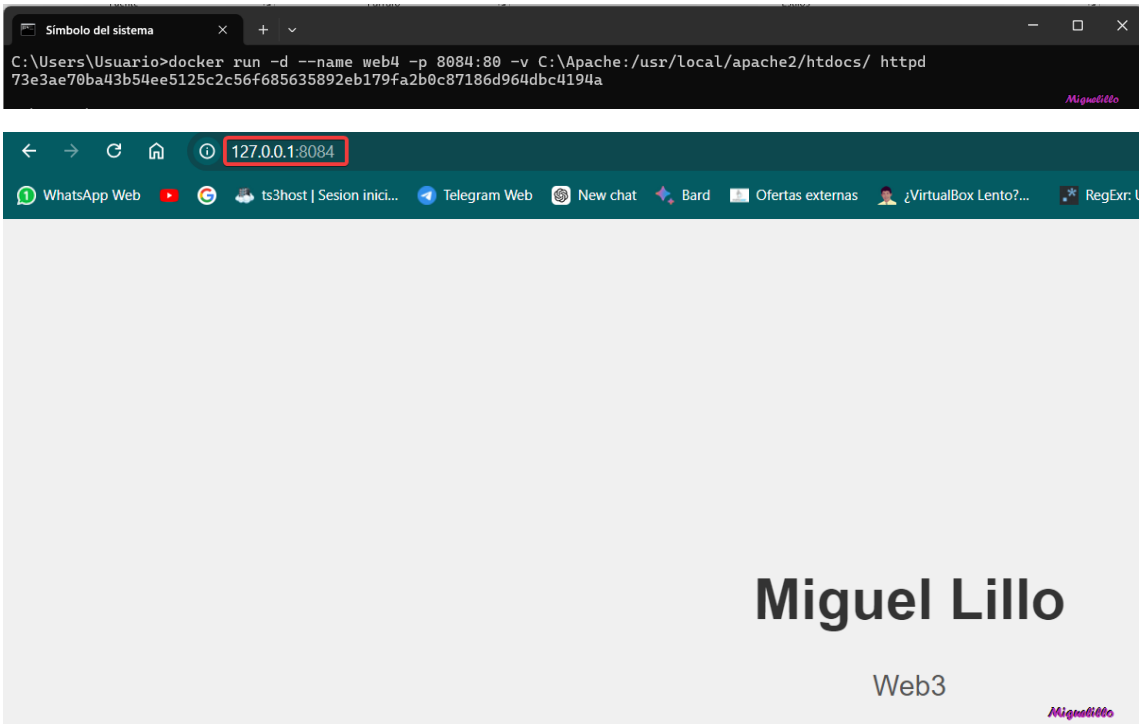
7) Ejecutar 4 veces más el contenedor con las mismas características que el caso anterior, en cada caso a los puertos 8041, 8084 y con los nombres web2..web5

Solución:

Creamos los diferentes puertos el 8041



El puerto 8084

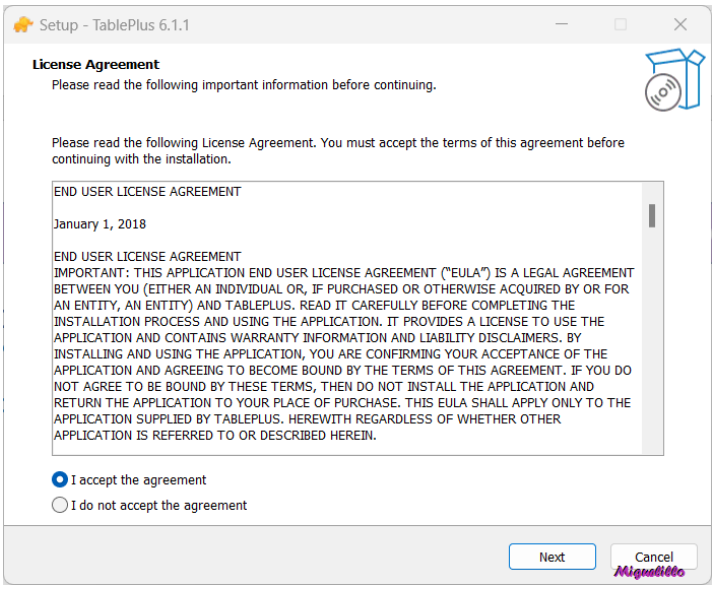


Y repetimos son las otras 2 páginas

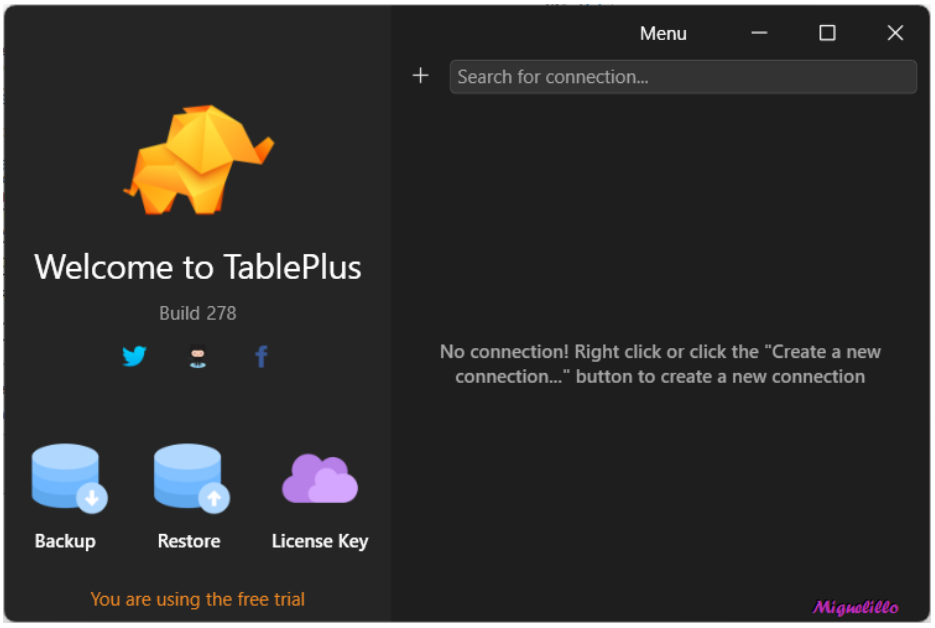
8) PostgreSQL

Solución:

Instalamos TablePlus

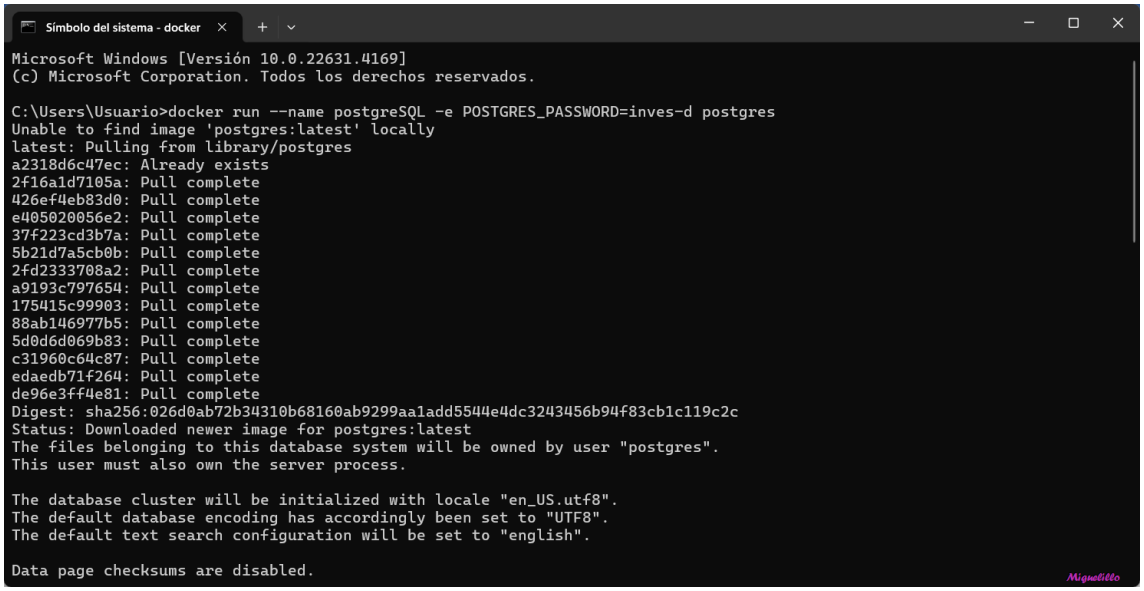


Se ha instalado correctamente, este nos servirápara conectarnos al postgre

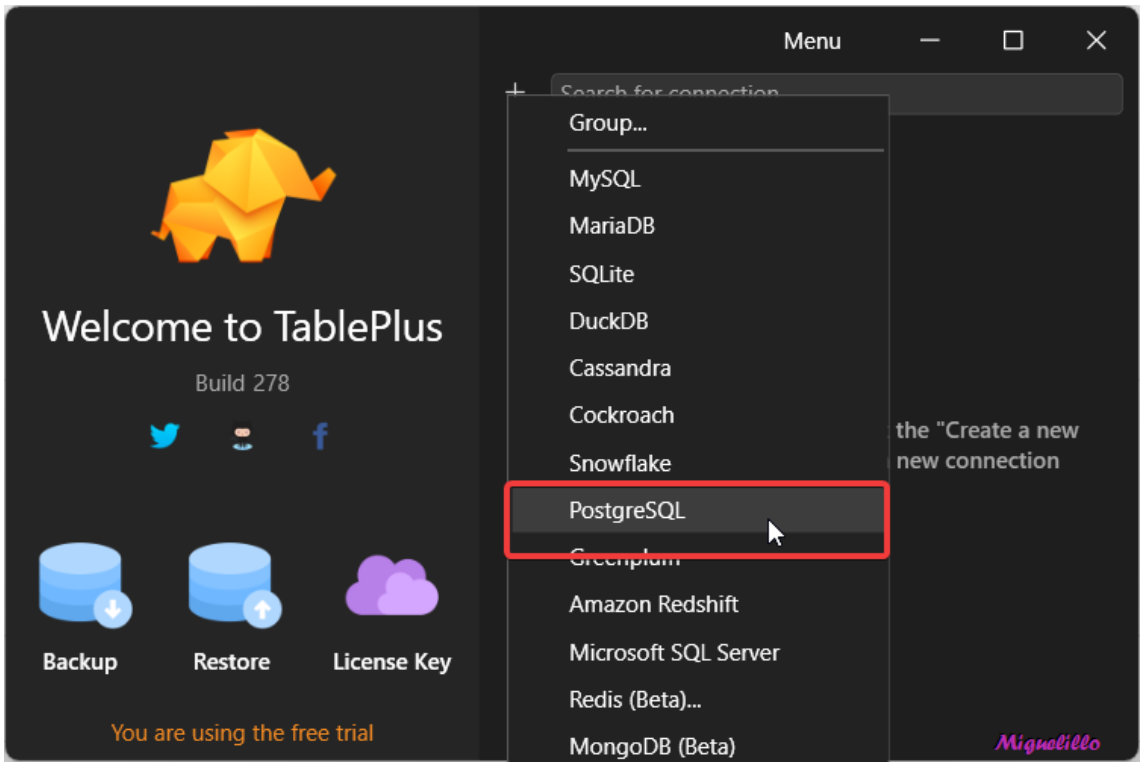


Instalamos PostgreSQL

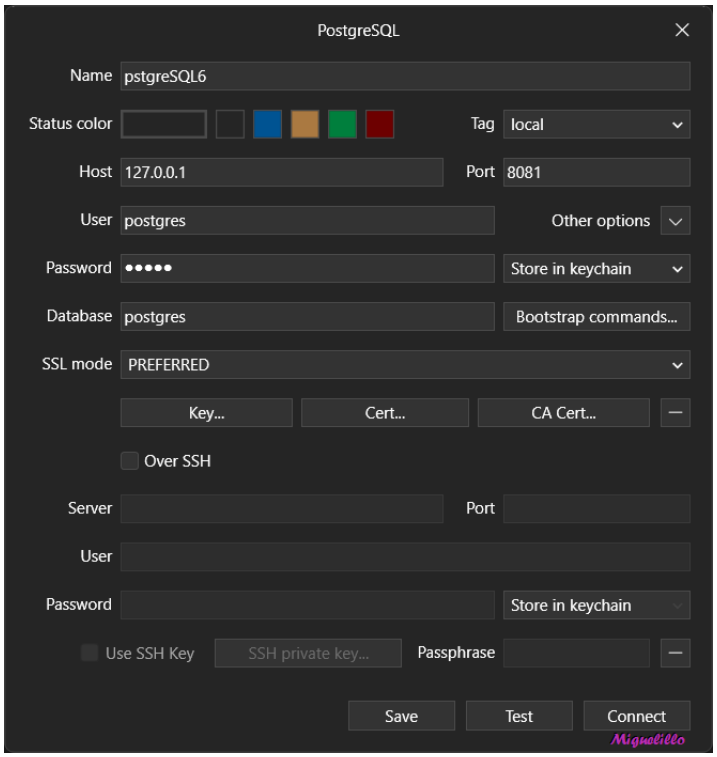
```
docker run --name postgreSQL -e POSTGRES_PASSWORD=inves -p 8081:5432 -d postgres
```



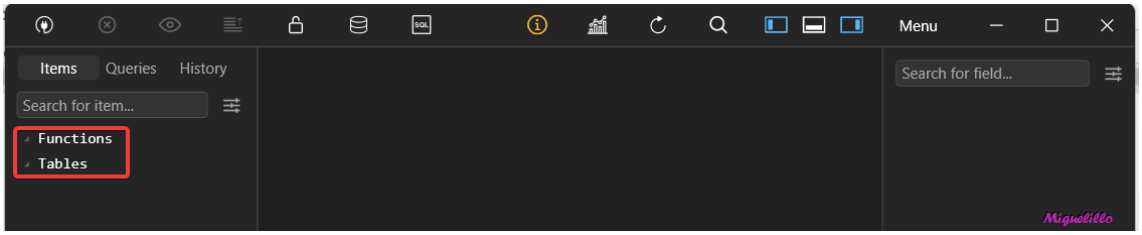
Para conectarnos



Añadimos los datos



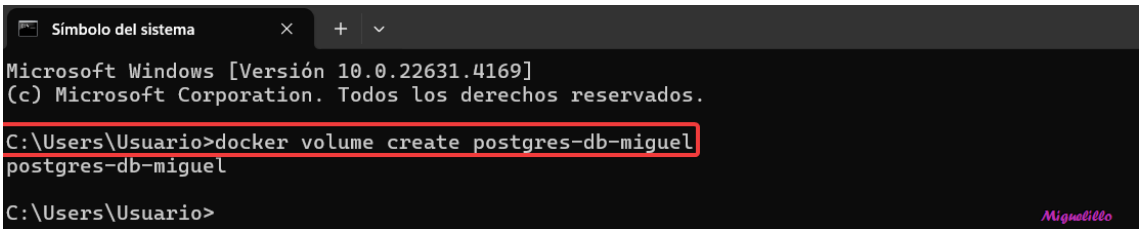
Probamos a conectarnos



Ejercicio 2

9) Crear un volumen para almacenar la información de la base de datos

Solución:

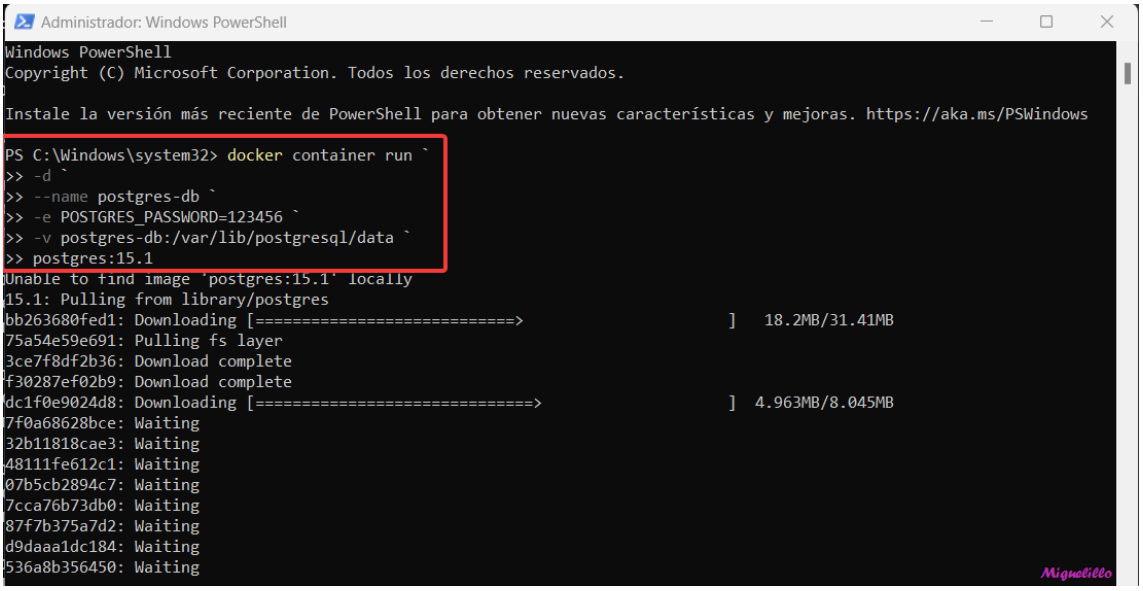


10) Montar la imagen de postgres así

Solución:

Comando

```
docker container run `
-d `
--name postgres-db `
-e POSTGRES_PASSWORD=123456 `
-v postgres-db:/var/lib/postgresql/data `
postgres:15.1
```

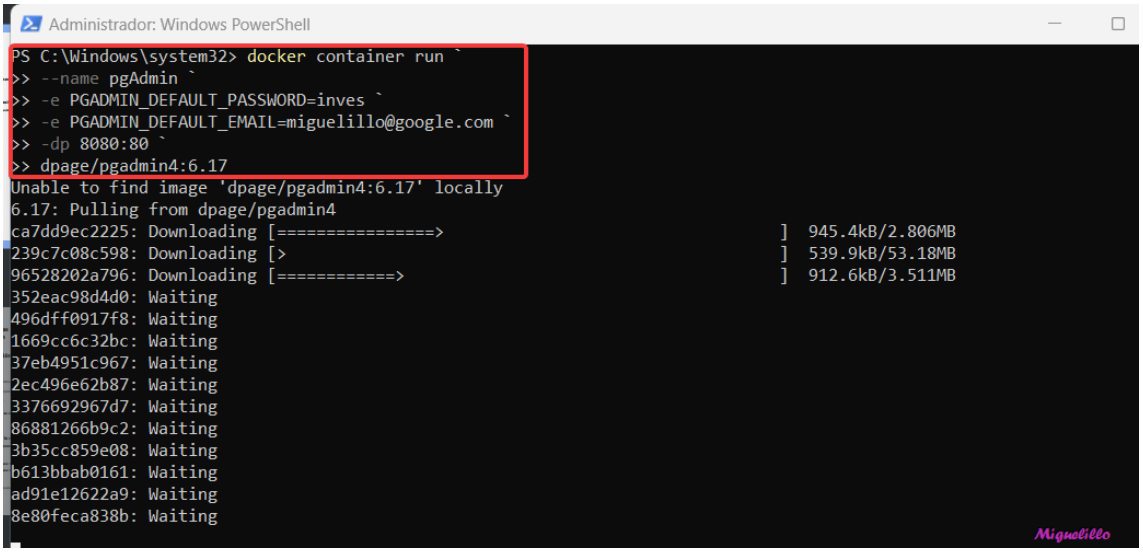


11) Tomar pgAdmin de aquí

Solución:

Instalamos pgAdmin

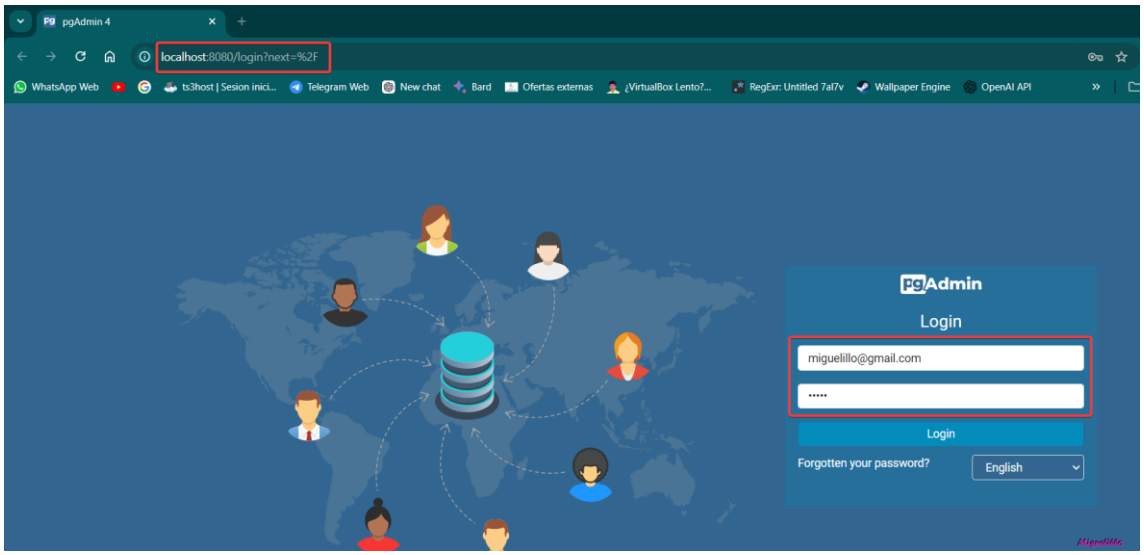
```
docker container run `
--name pgAdmin `
-e PGADMIN_DEFAULT_PASSWORD=inves `
-e PGADMIN_DEFAULT_EMAIL=miguelillo@google.com `
-dp 8080:80 `
dpage/pgadmin4:6.17
```



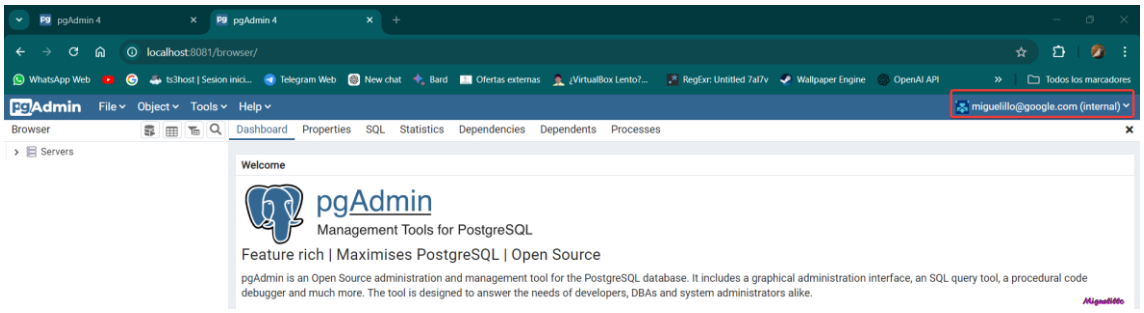
12) Ingresar a la web con las credenciales de superman

Solución:

Vamos a la web y nos logueamos

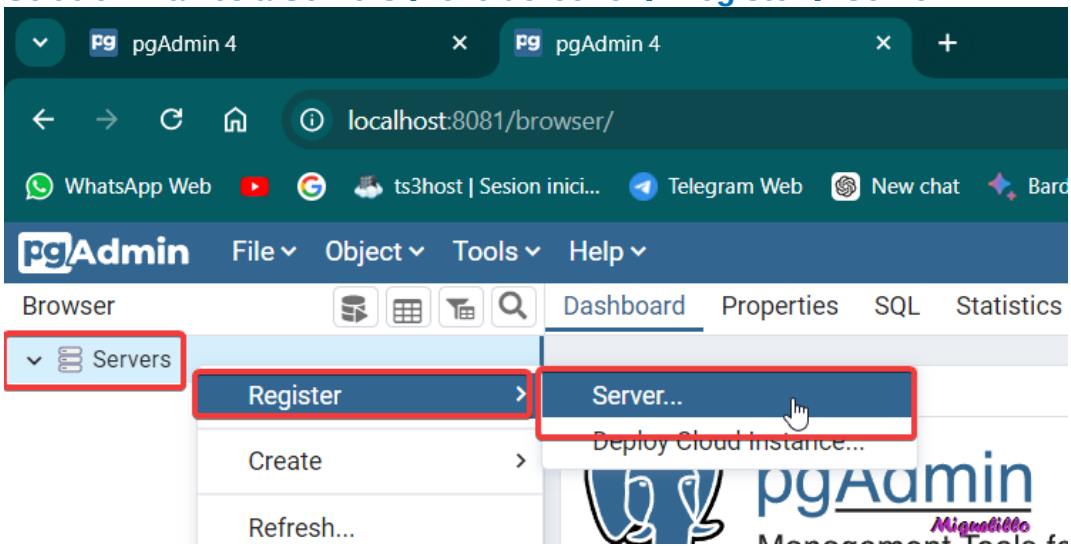


Hemos iniciado sesión

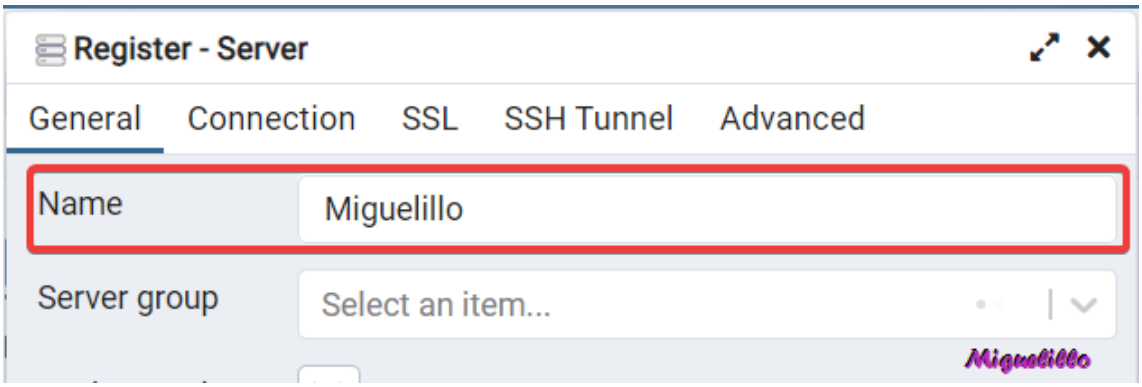


13) Intentar crear la conexión a la base de datos

Solución: Vamos a **Servers→ clic derecho → Register→ Server…**



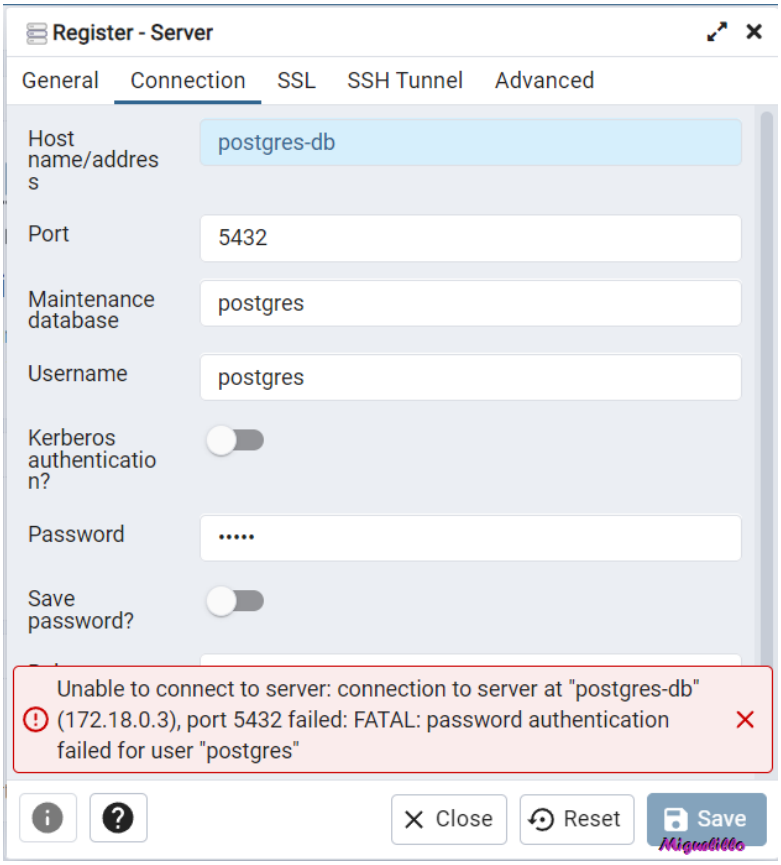
Le ponemos el nombre

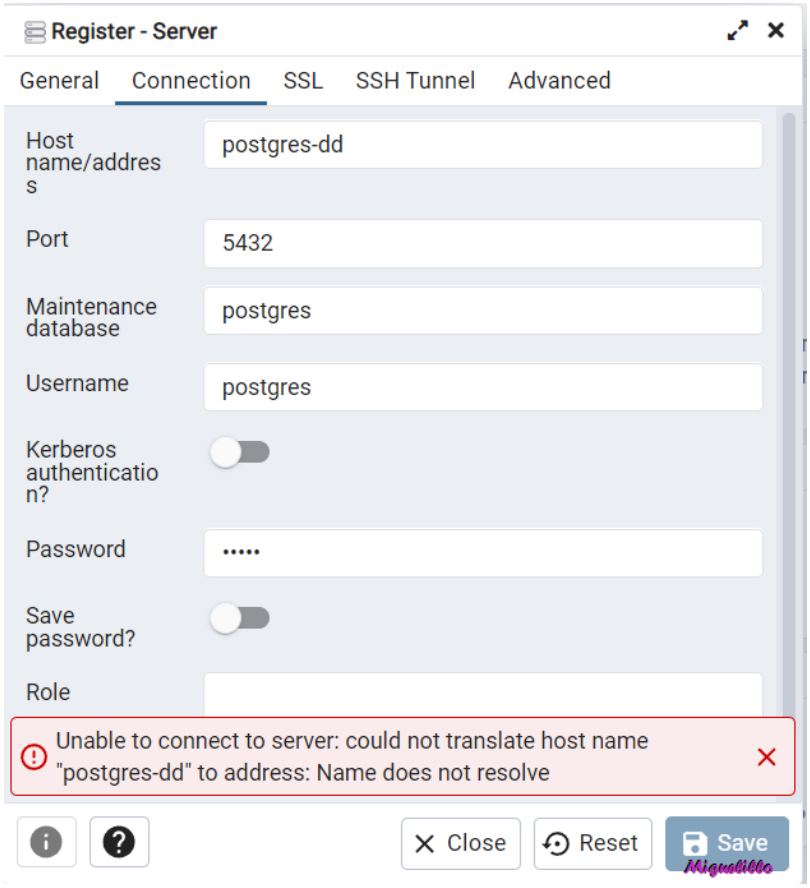


14) Ohhh no!, no vemos la base de datos, se nos olvidó la red

Solución:

Nos da errores como los siguientes

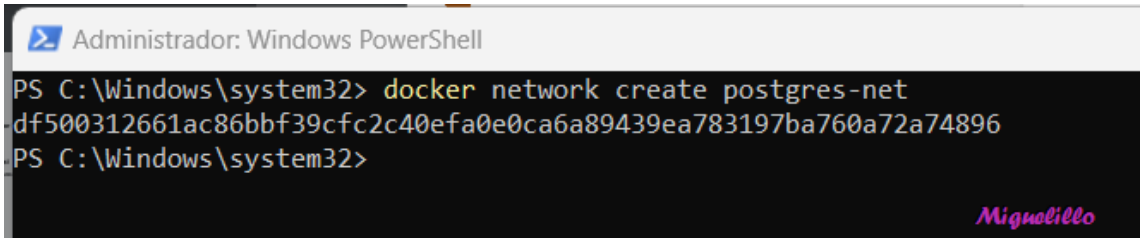




15) Crear la red

Solución:

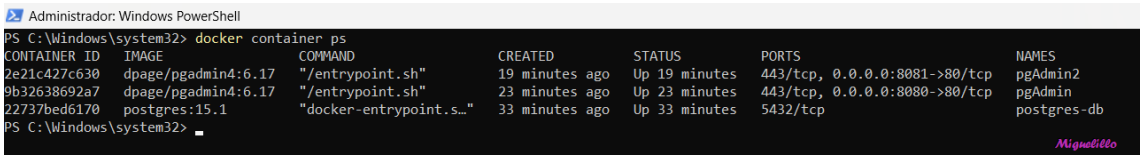
```
docker network create postgres-net
```



16) Asignar ambos contenedores a la red

Solución:

```
docker container ps
```

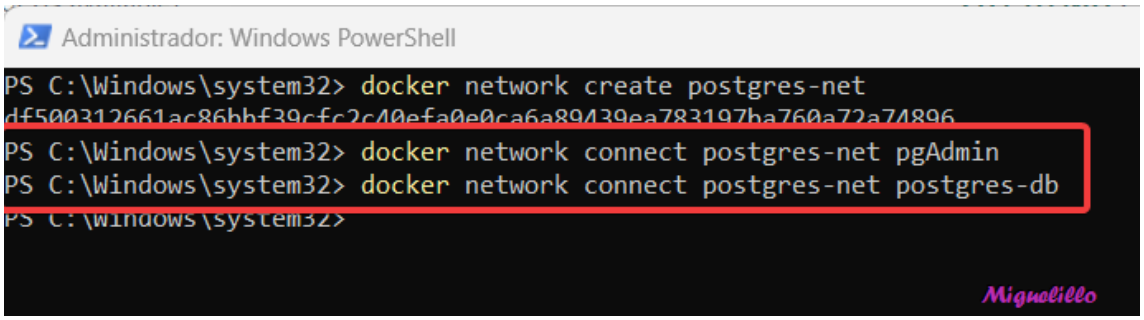


17) Conectar ambos contenedores

Solución:

```
docker network connect postgres-net
```

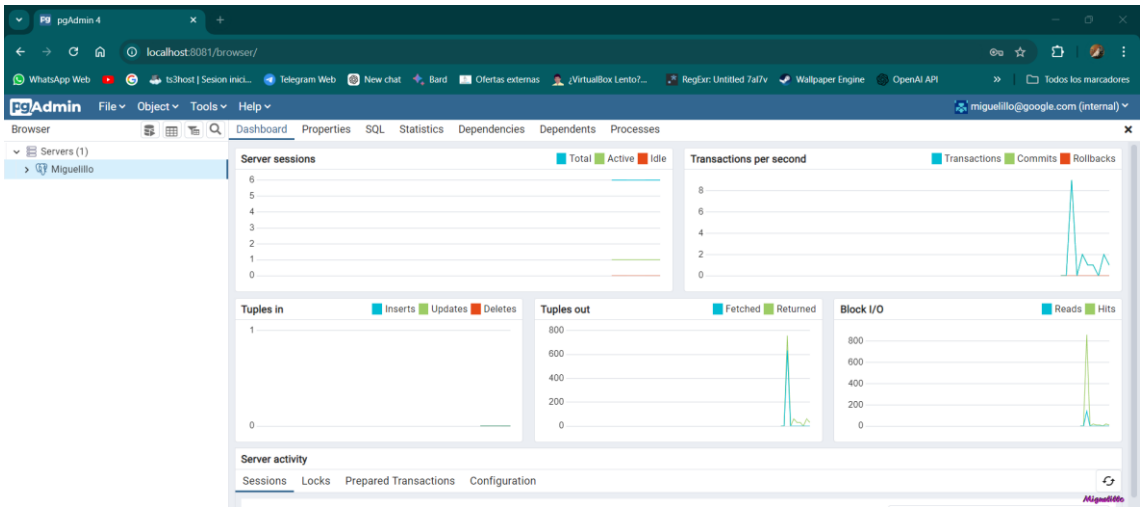
```
docker network connect pgAdmin
```



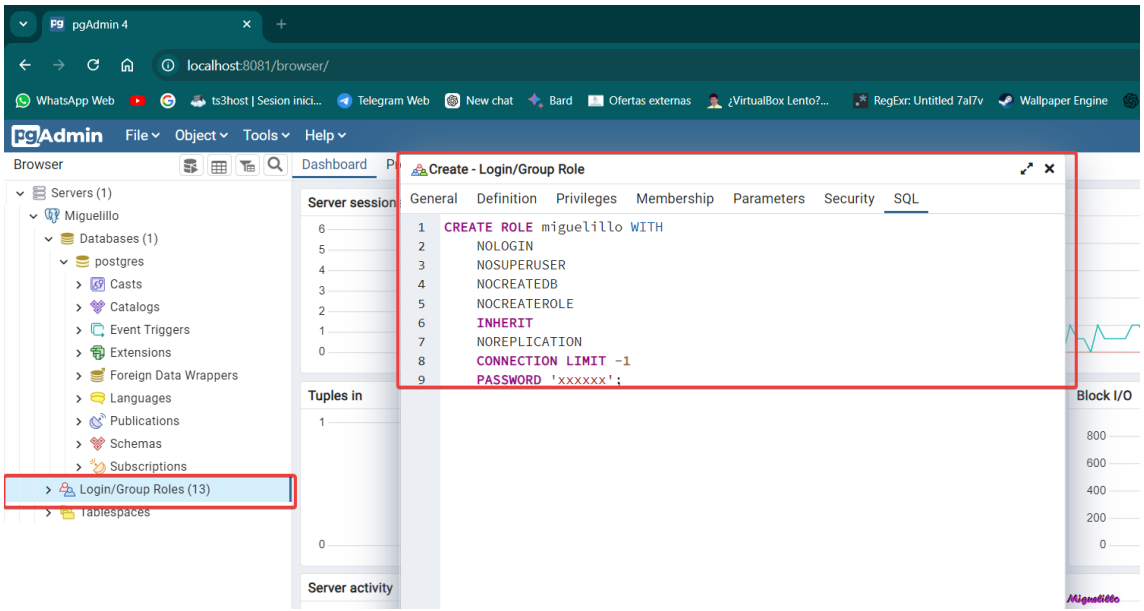
18) Intentar el paso 4. de nuevo.

Solución:

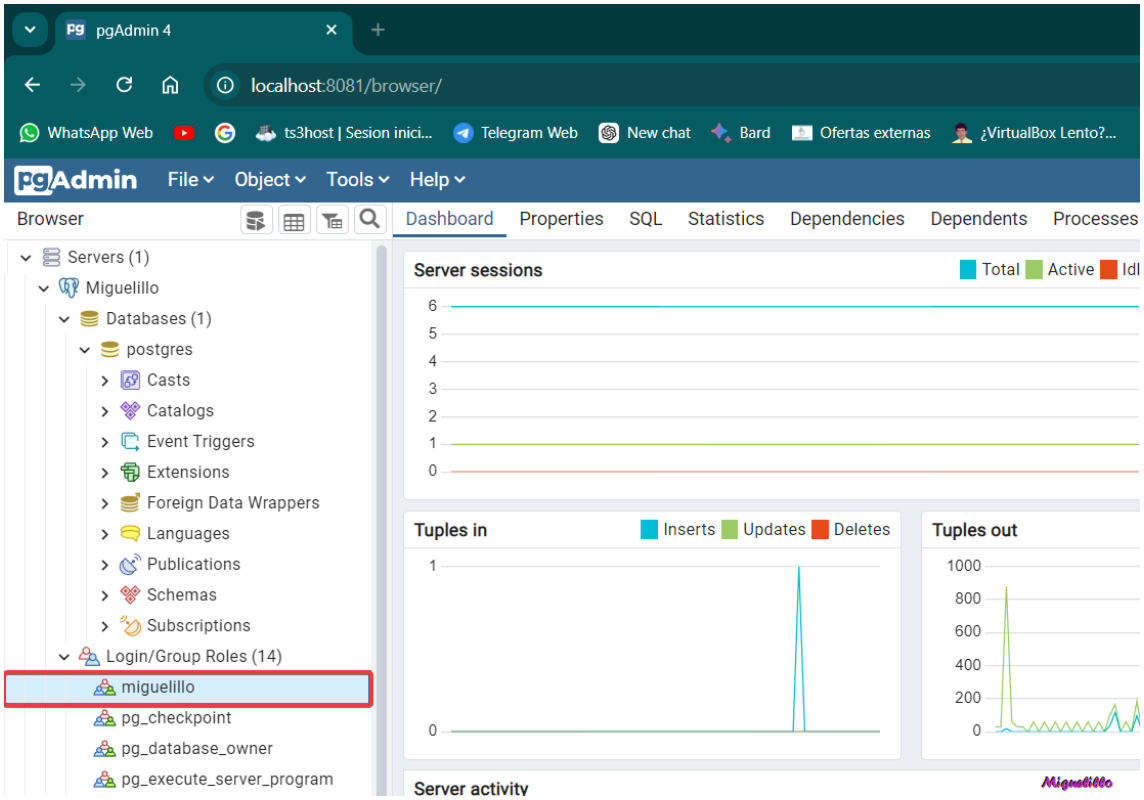
Se conecta correctamente



Creamos un usuario



Se ha creado el usuario



19) Qué es Docker Compose

Docker Compose es una utilidad de Docker que permite ejecutar varios contenedores simultáneamente reuniéndolo en un solo contenedor

20) Crear Docker Compose

Creamos un Docker Compose, este reúne todos los pasos anteriormente

51-55 pgAdmin.yaml

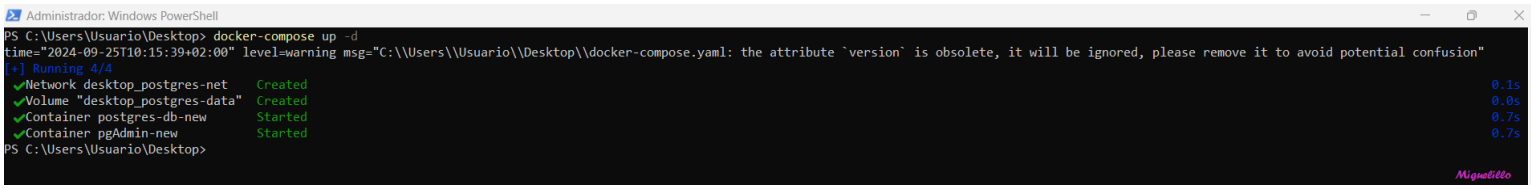
Archivo Editar Ver

```
version: '3'
services:
  postgres-db:
    image: postgres:15.1
    container_name: postgres-db-new
    environment:
      POSTGRES_PASSWORD: 123456
    volumes:
      - postgres-data:/var/lib/postgresql/data
    networks:
      - postgres-net

  pgadmin:
    image: dpage/pgadmin4:6.17
    container_name: pgAdmin-new
    environment:
      PGADMIN_DEFAULT_EMAIL: superman@google.com
      PGADMIN_DEFAULT_PASSWORD: 123456
    ports:
      - "8080:80"
    networks:
      - postgres-net

volumes:
  postgres-data:

networks:
  postgres-net:
```

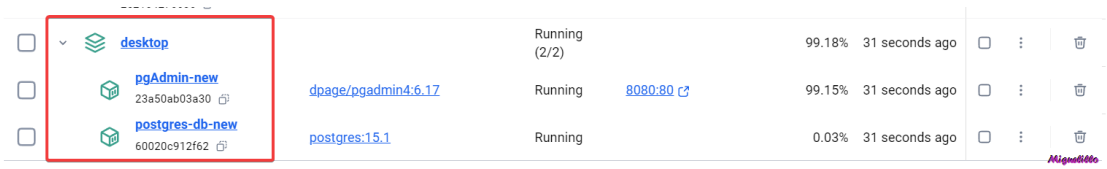
Miguelillo

Probamos el docker Compose vamos a la ruta donde lo tenemso alamacenado y lo ejecutamos con

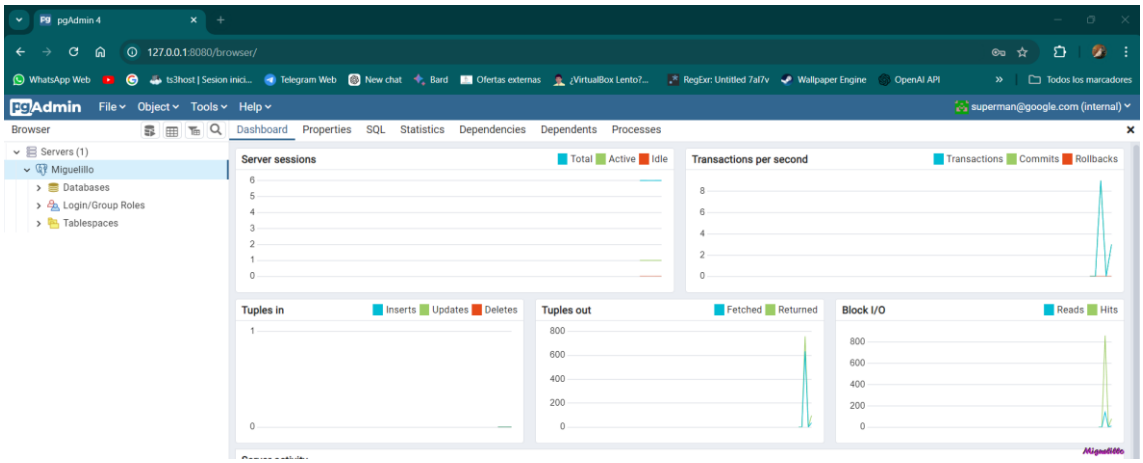
```
docker-compose up -d
```



Se ha creado correctamente



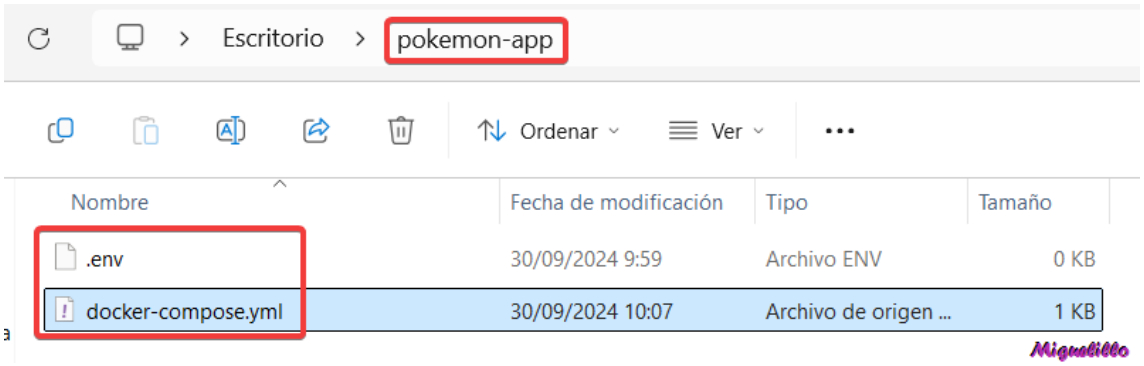
Funciona correctamente


pgAdmin 4
127.0.0.1:8080/browser/
pgAdmin
File
Object
Tools
Help
superman@google.com (internal)
Browser
Servers (1)
Miguelillo
Databases
Login/Group Roles
Tablespaces
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
Server sessions
Total
Active
Idle
Transactions per second
Transactions
Commits
Rollbacks
Tuples in
Inserts
Updates
Deletes
Tuples out
Fetched
Returned
Block I/O
Reads
Hits

Ejercicio 3

21) Cread una capeta que se llame “pokemon-app”

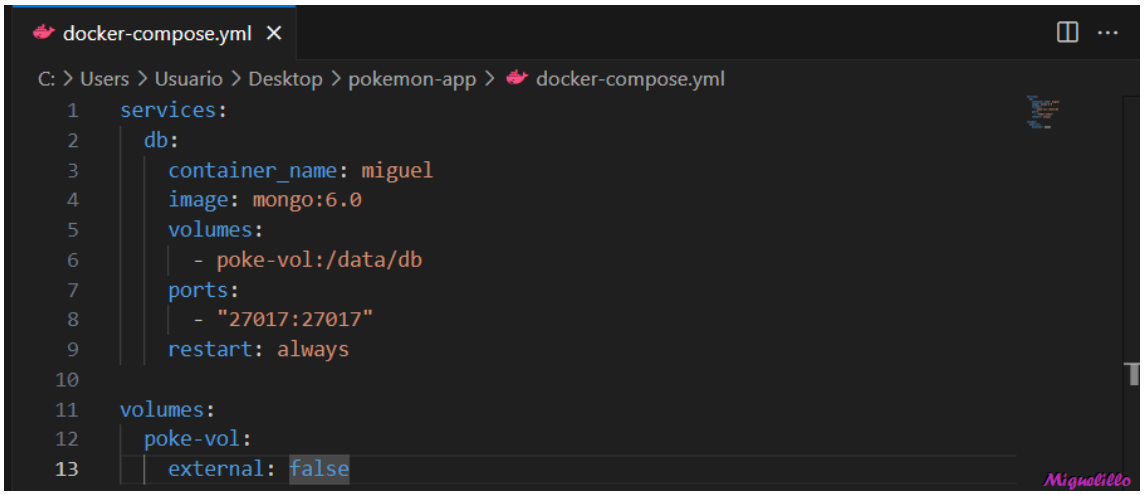
Solución:



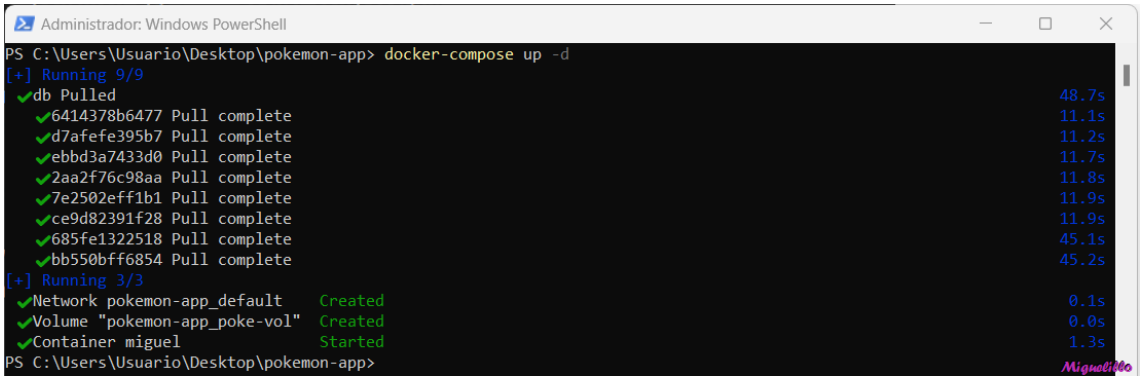
22) Escribir en nuestro docker-compose

Solución:

Editamos el archivo



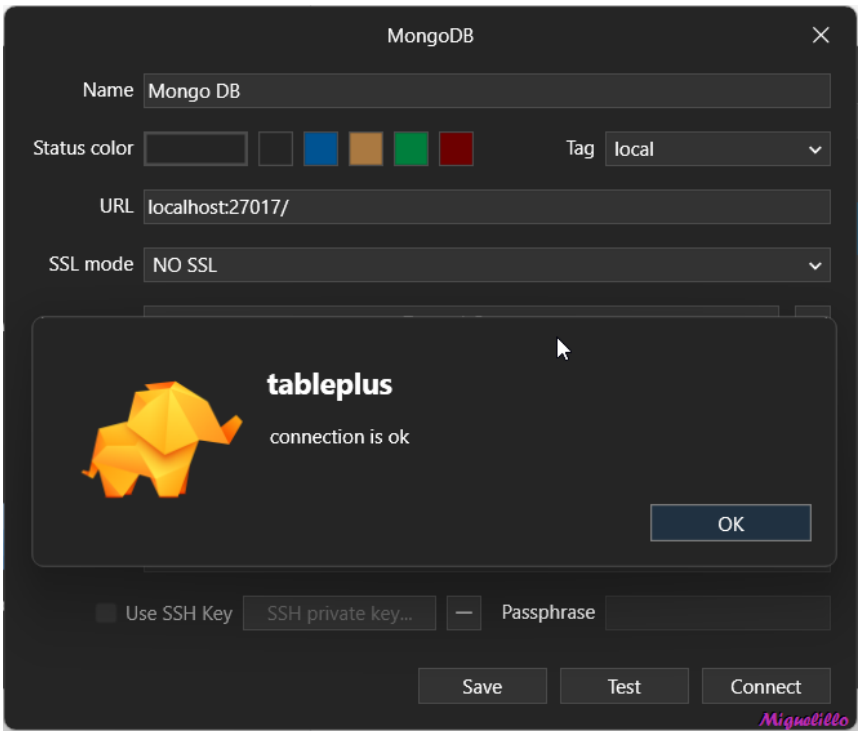
Lo ejecutamos



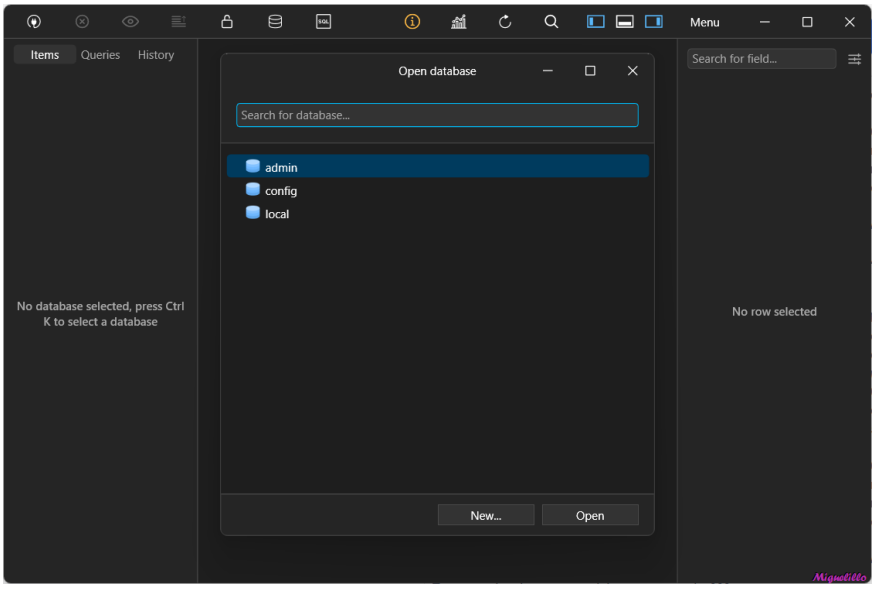
23) Probar la conexión en TablePlus seleccionando mongo

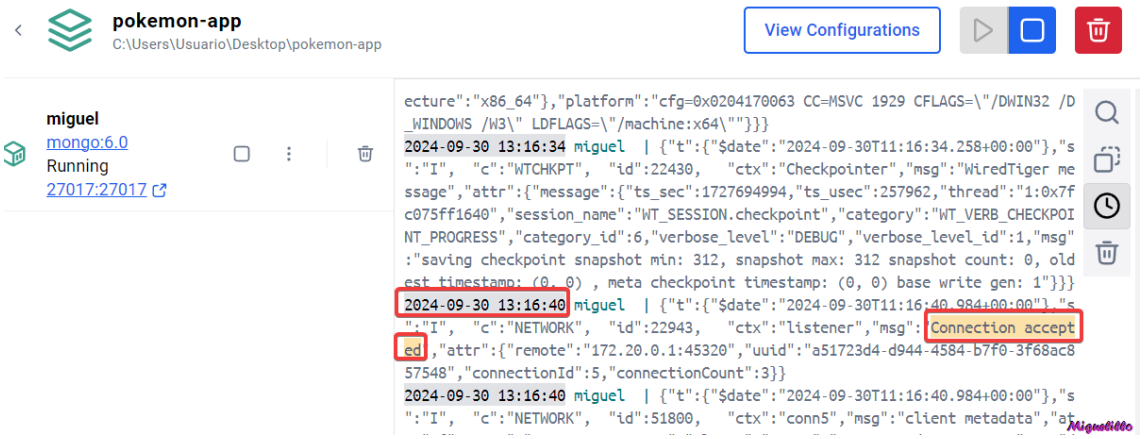
Solución:

Probamos la conexión y es exitosa



Vemos que accede correctamente

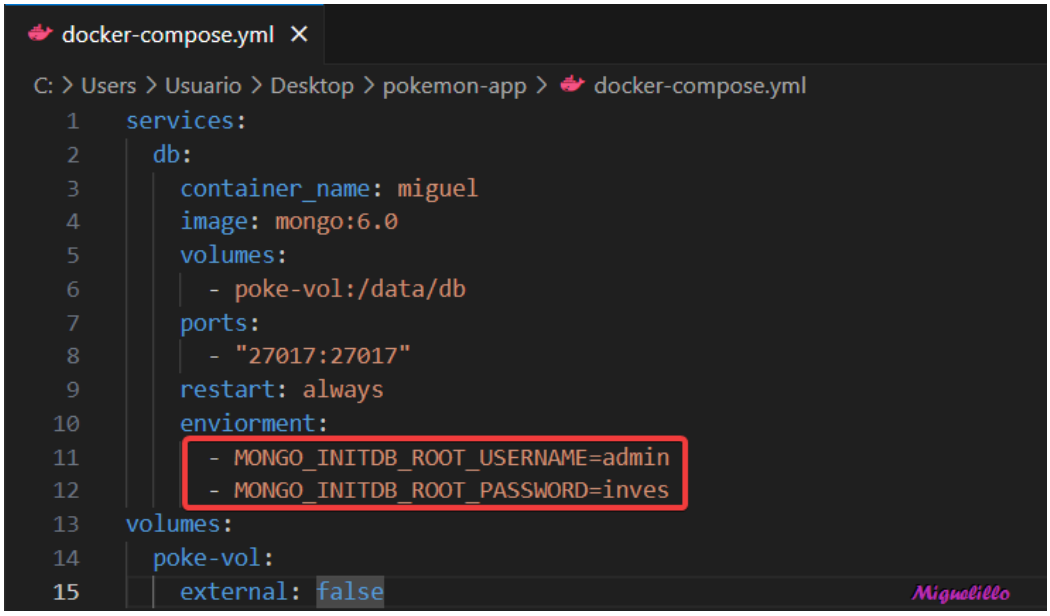



pokemon-app
C:\Users\Usuario\Desktop\pokemon-app
View Configurations
miguel
mongo:6.0
Running
27017:27017
ecture":"x86_64"},"platform":"cfg=0x0204170063 CC=MSVC 1929 CFLAGS=\"/DWIN32 /D_WINDOWS /W3\" LDFLAGS=\"/machine:x64\""}}}
2024-09-30 13:16:34 miguel | {"t":{"$date":"2024-09-30T11:16:34.258+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpointer","msg":"WiredTiger message","attr":{"message":{"ts_sec":1727694994,"ts_usec":257962,"thread":"1:0x7fc075ff1640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","category_id":6,"verbose_level":"DEBUG","verbose_level_id":1,"msg":"saving checkpoint snapshot min: 312, snapshot max: 312 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 1"}}}
2024-09-30 13:16:40 miguel | {"t":{"$date":"2024-09-30T11:16:40.984+00:00"},"s":"I", "c":"NETWORK", "id":22943, "ctx":"listener","msg":"Connection accepted","attr":{"remote":"172.20.0.1:45320","uuid":"a51723d4-d944-4584-b7f0-3f68ac857548","connectionId":5,"connectionCount":3}}
2024-09-30 13:16:40 miguel | {"t":{"$date":"2024-09-30T11:16:40.984+00:00"},"s":"I", "c":"NETWORK", "id":51800, "ctx":"conn5","msg":"client metadata","at

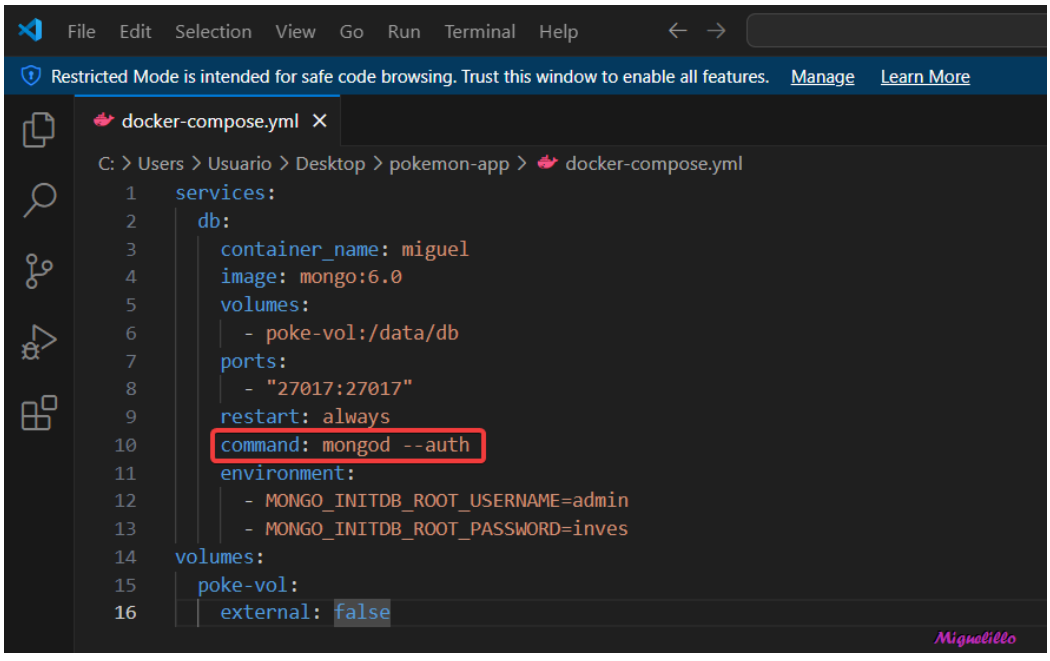
24) Ahora queremos que las conexiones al mongo estén autenticadas

Solución:

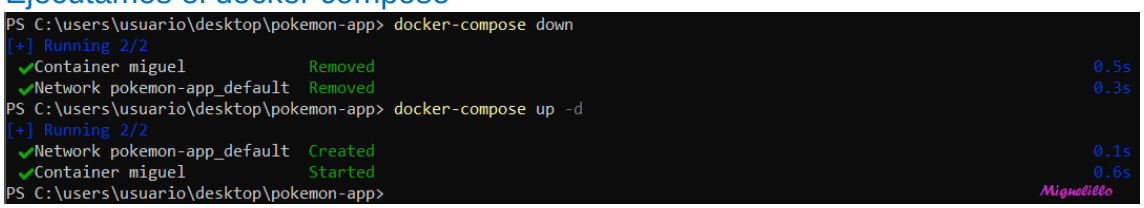
Añadimos las variables para introducir el usuario y la contraseña



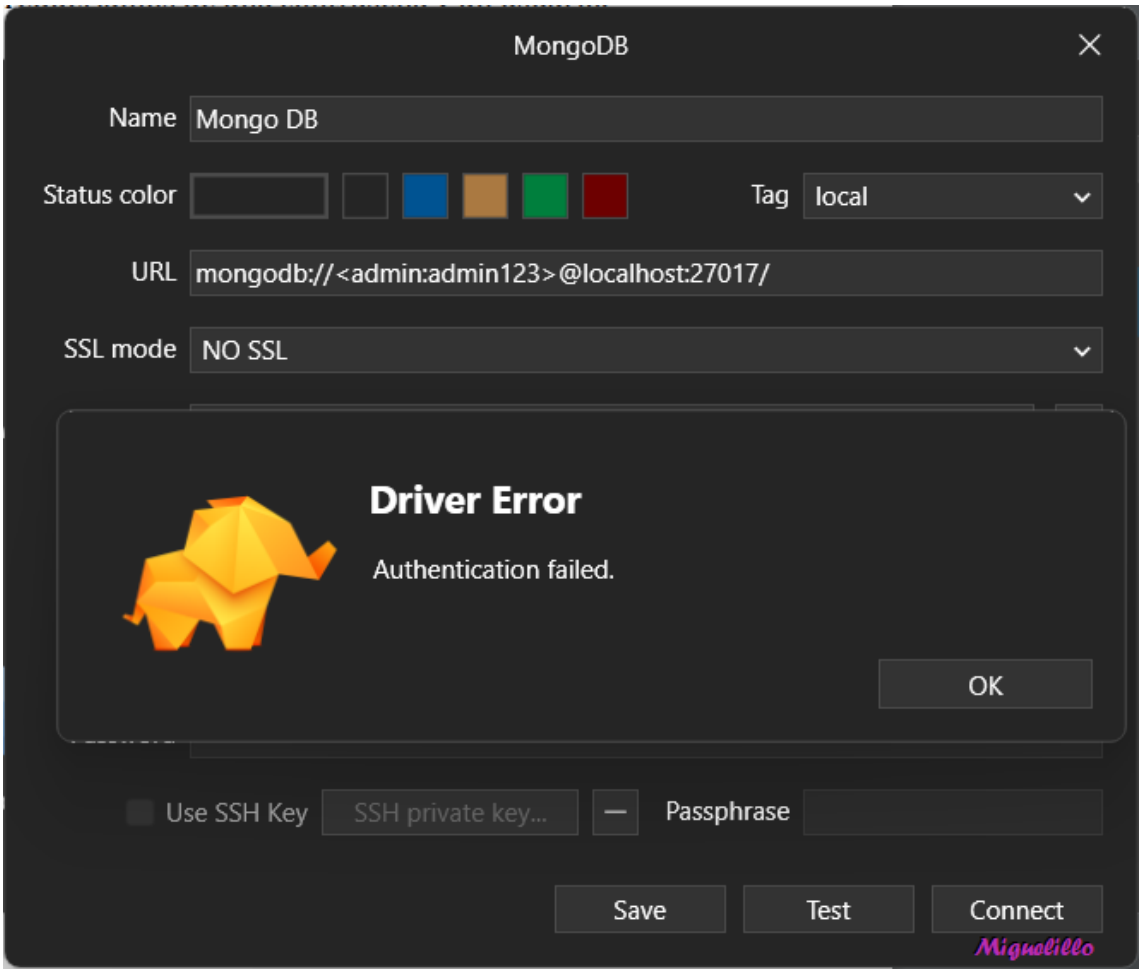
Añadimos la siguiente opción para que nos pida la autenticación



Ejecutamos el docker compose



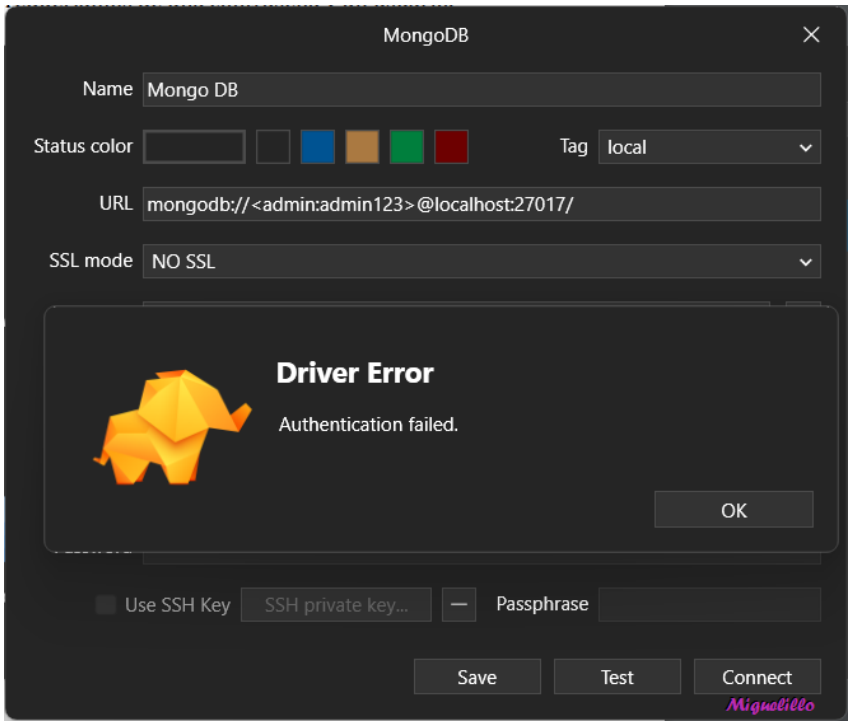
Probamos a conectarnos y nos da el siguiente fallo



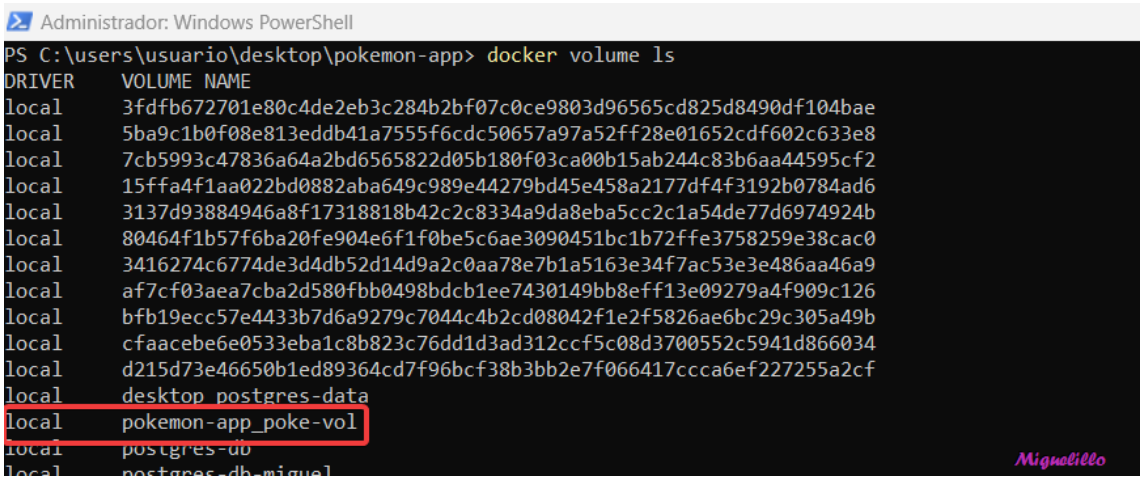
25) Toca editar la conexión para que nos deje

Solución:

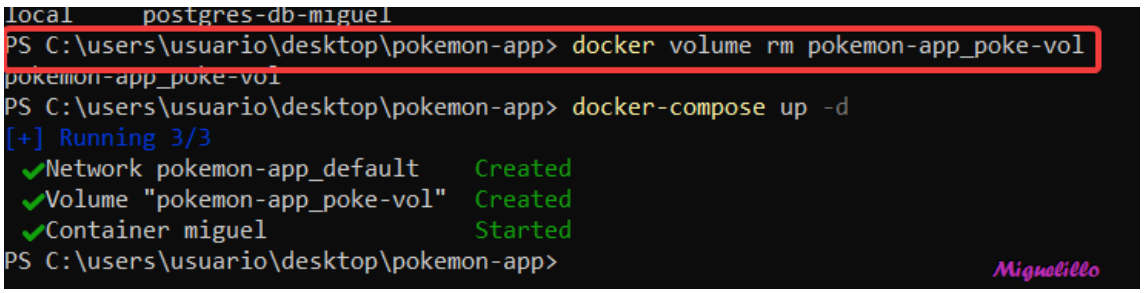
Fallo de autenticación



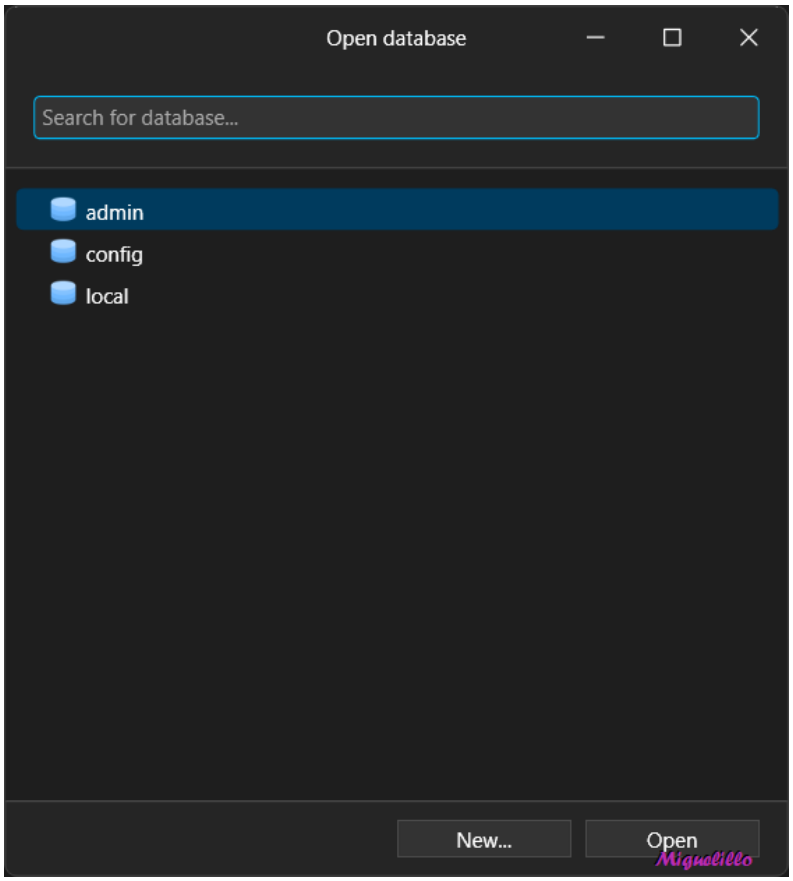
Listamos los volúmenes



Borramos el volumen



Nos podemos conectar correctamente

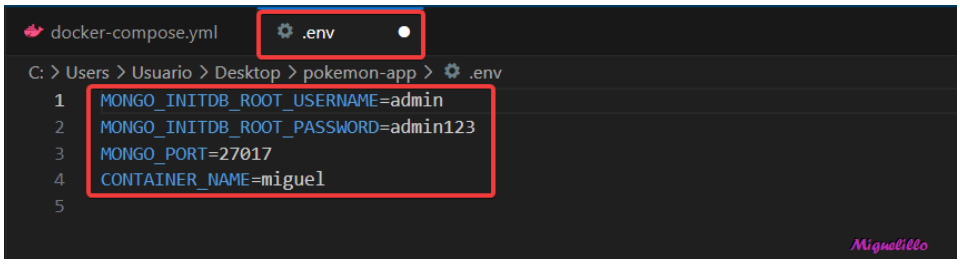


26) ARCHIVO .env

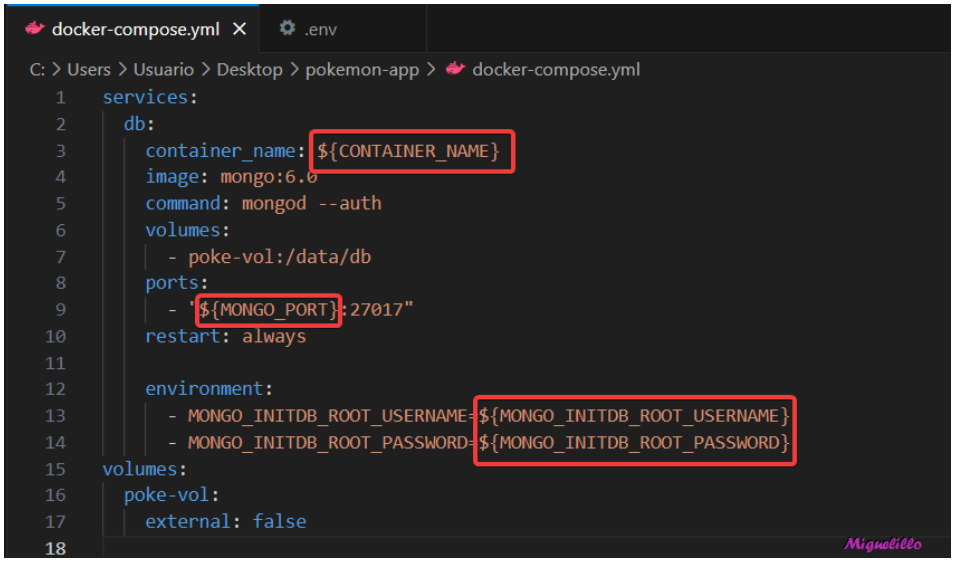
Solución:

Un archivo .**env** es un **archivo** de **texto** que contiene **variables de entorno** y se utiliza para **configurar parámetros**

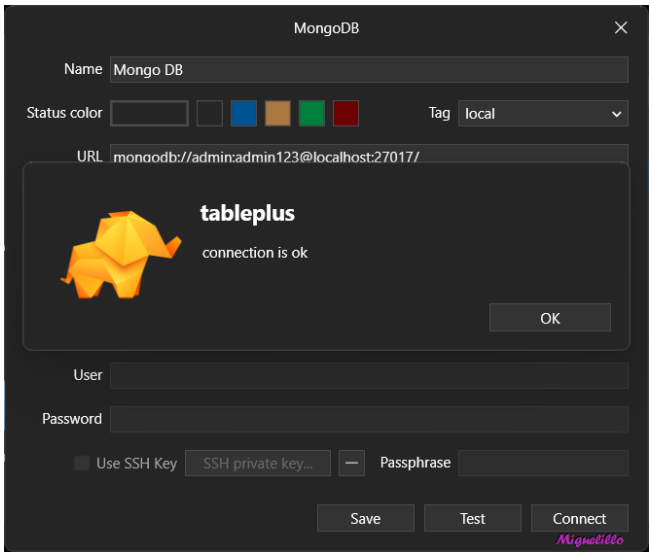
Editamos el .env



Editamos el yml

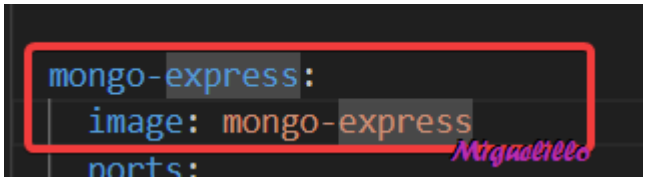


Sigue funcionando correctamente



27) Añadimos la imagen de mongo-express

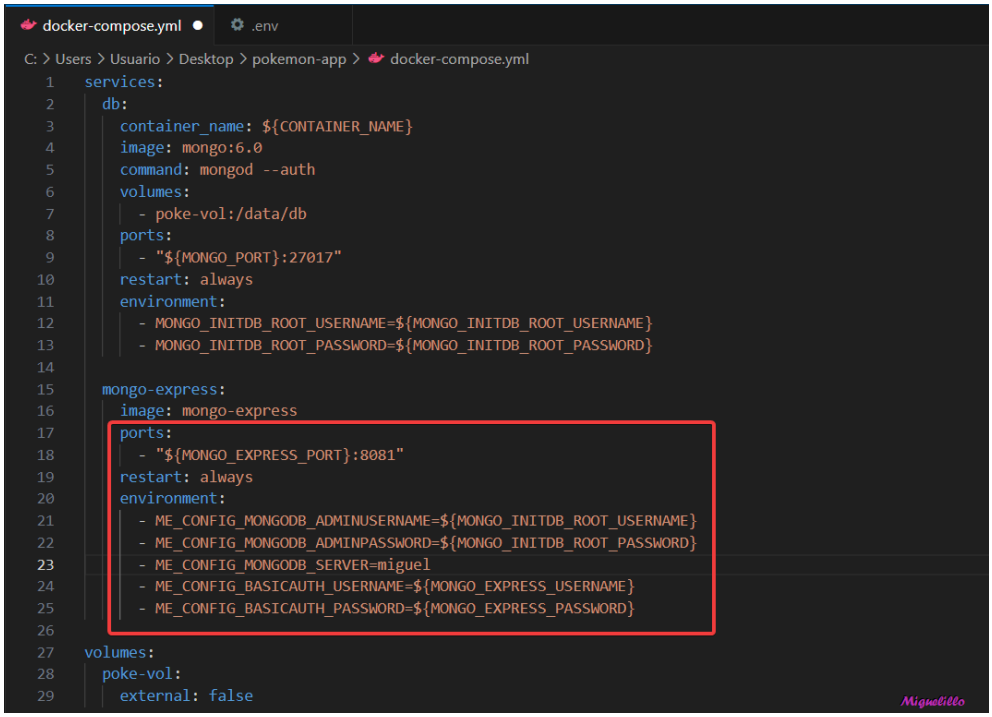
Solución:



28) Añadir a la imagen más variables de entorno, puertos y always

Solución:

Configuramos el usuario contraseña, puertos, el always y el servidor de mondgodb

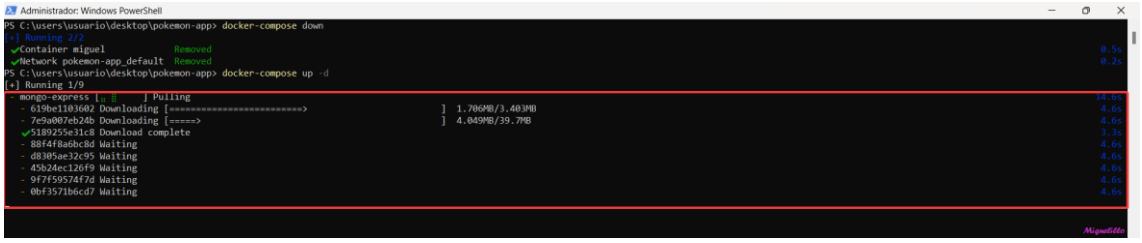


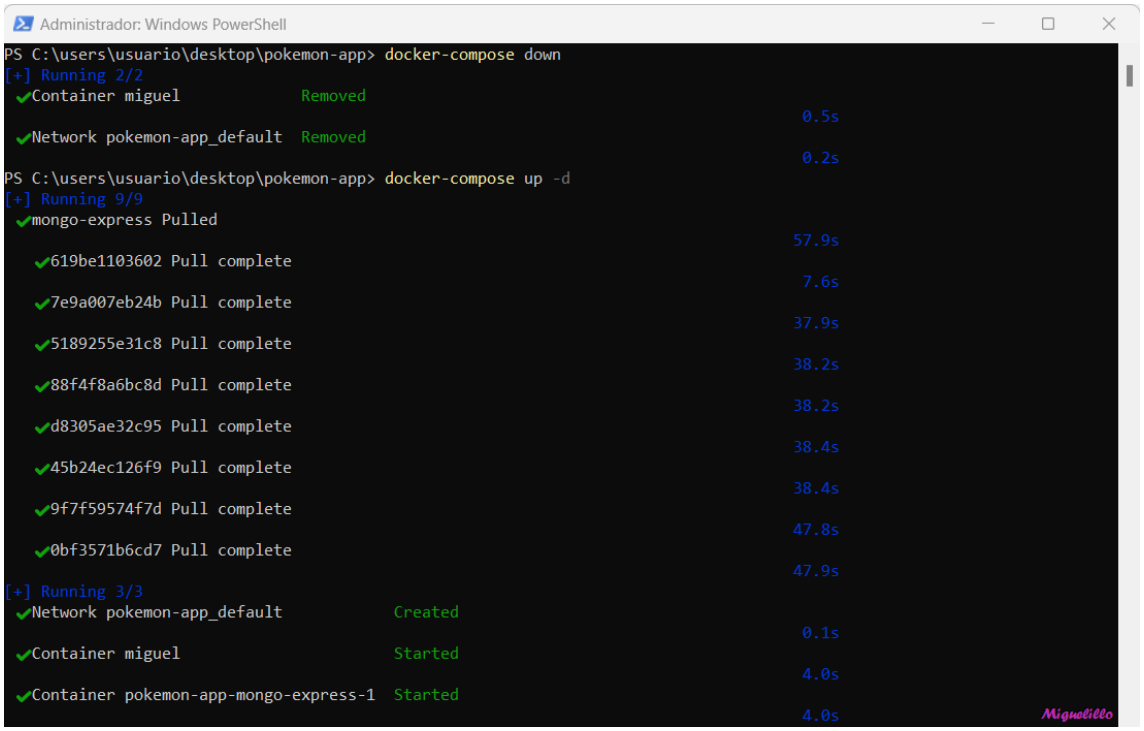
Configuramos las variables del .env

29) Levantamos el docker-compose

Solución:

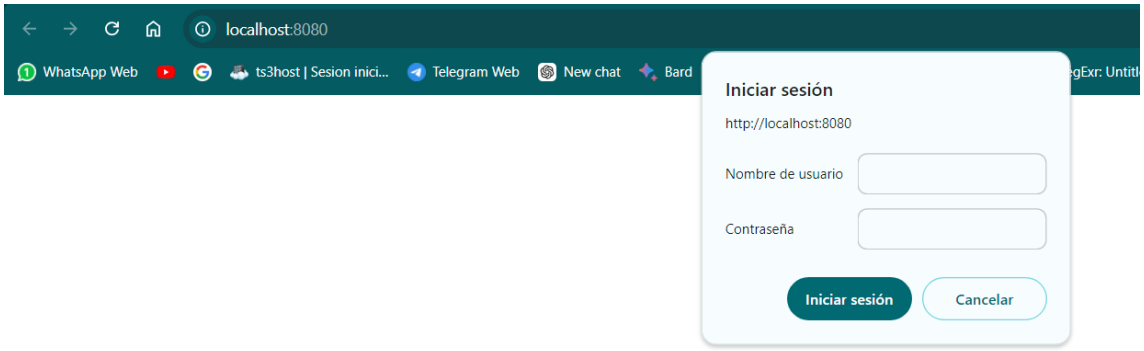
Se levanta correctamente



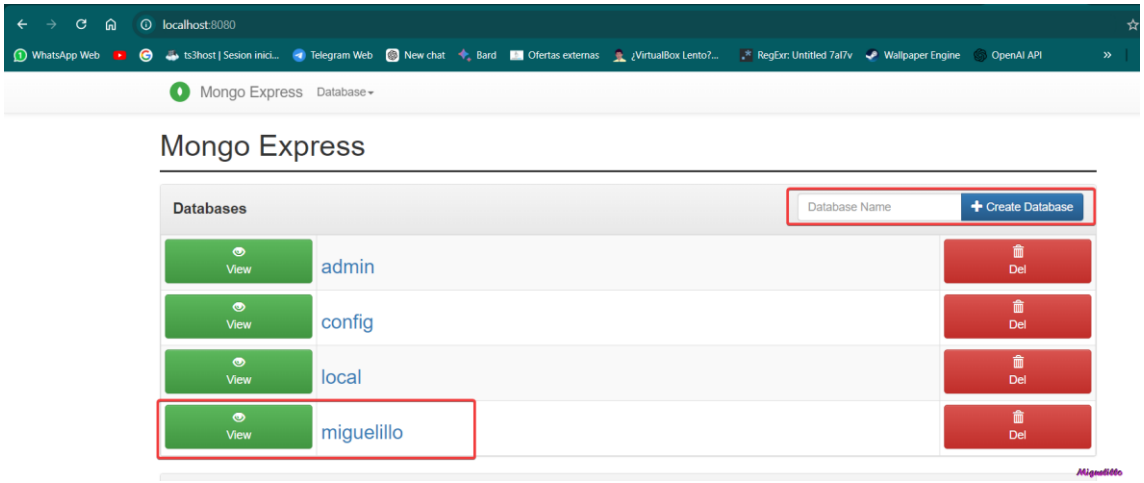


30) Probamos a acceder

Solución:



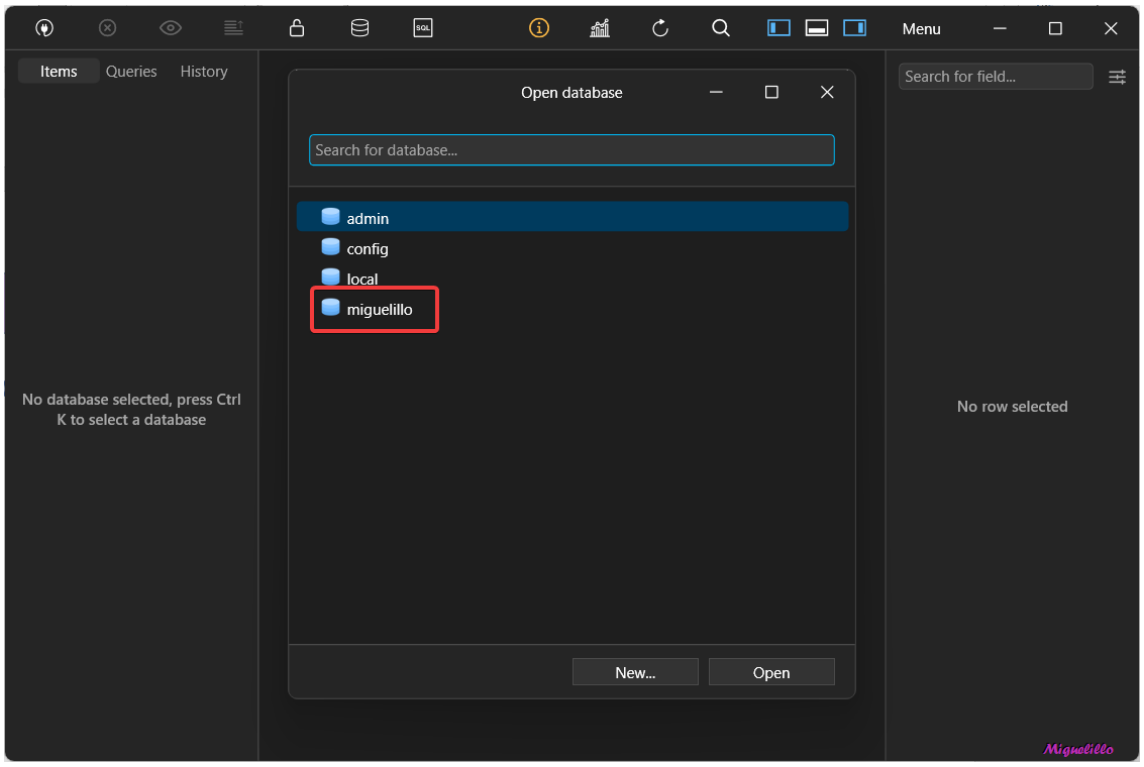
Creamos la base de datos “miguelillo”



31) Verificamos con table plus

Solución:

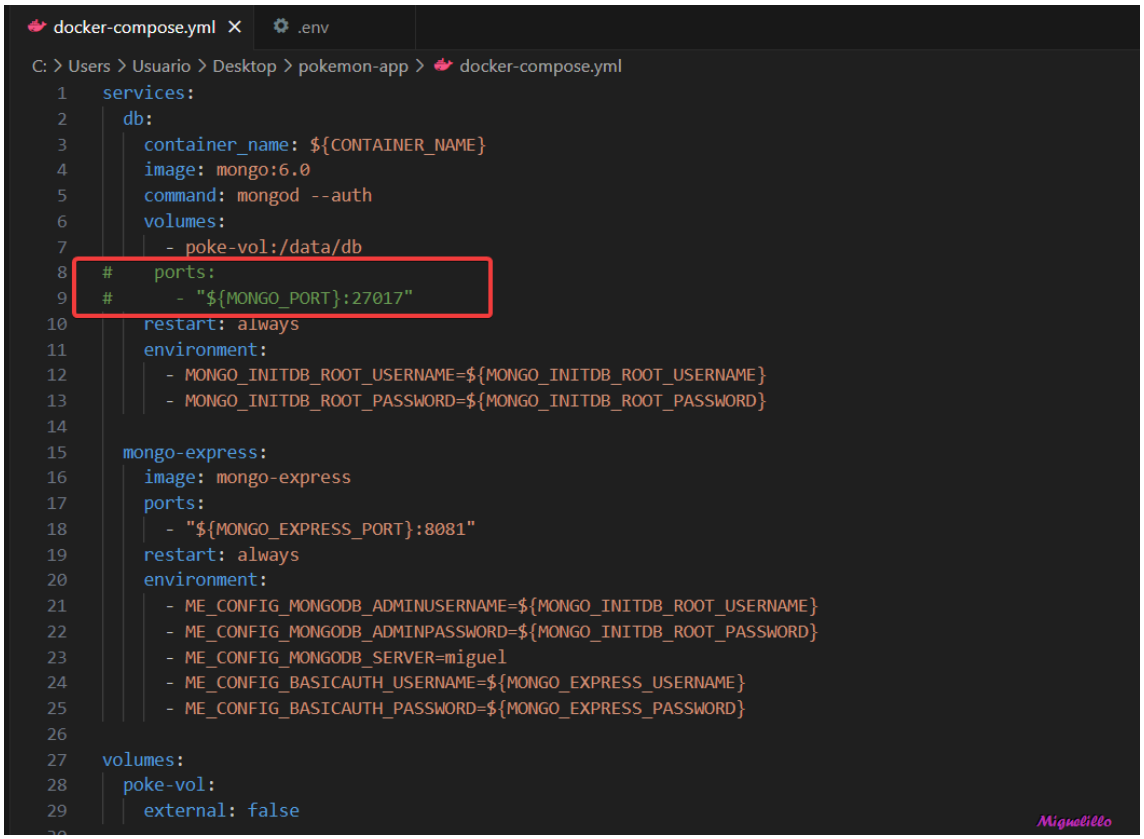
Se ve correctamente



32) Comentar los puertos de mongoDB

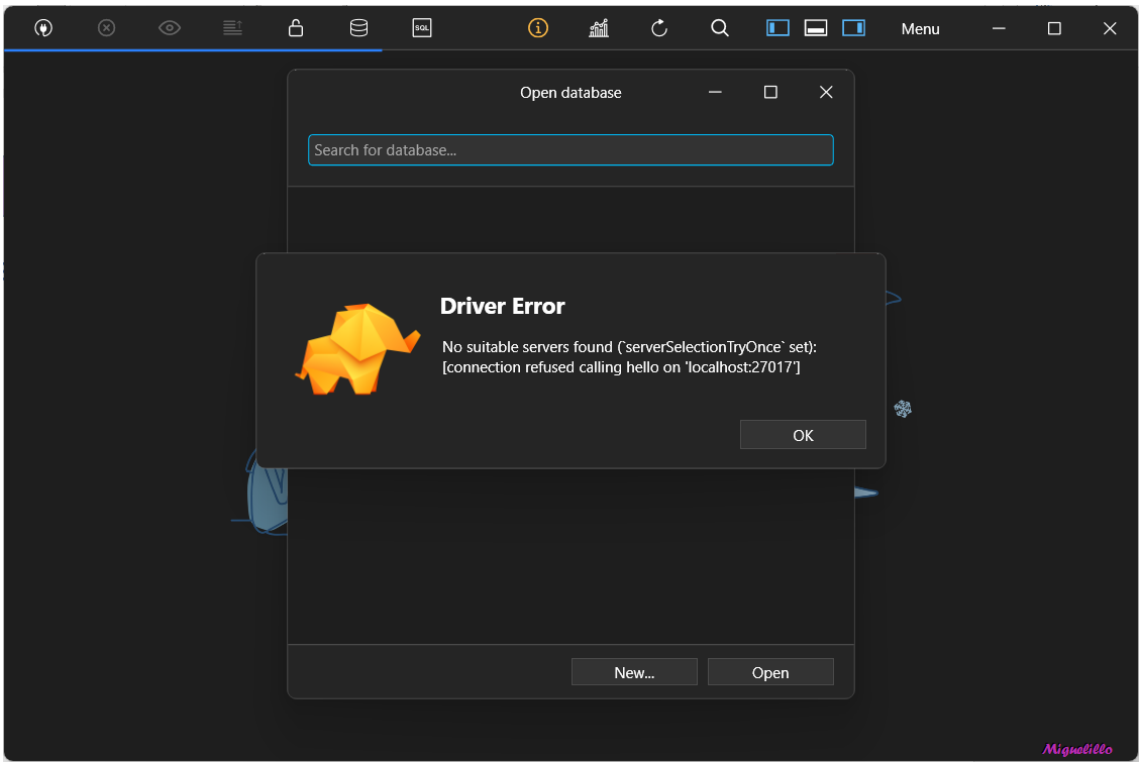
Solución:

Los comento añadiendo un

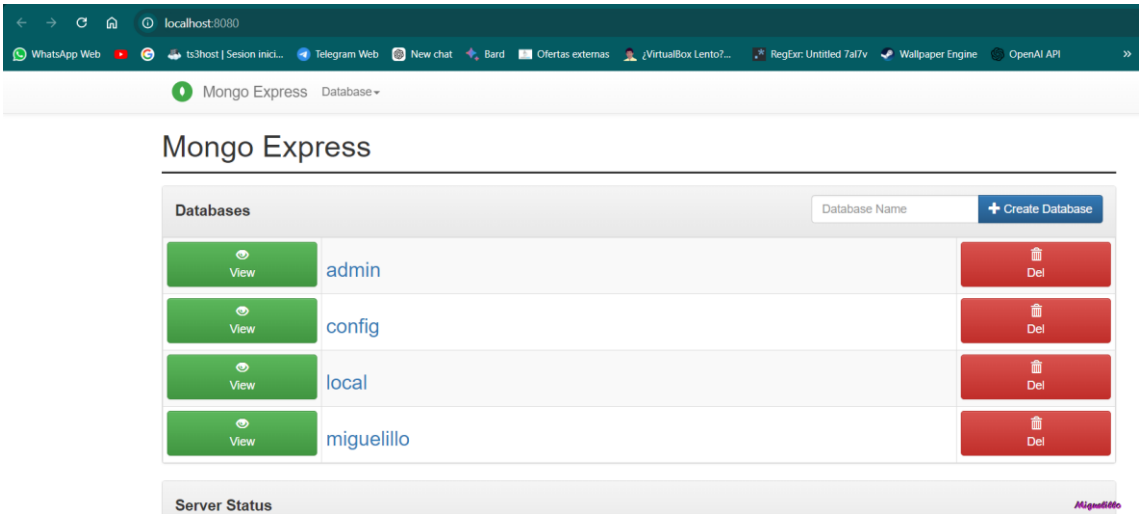


33) Probamos que sea inaccesible

Solución:



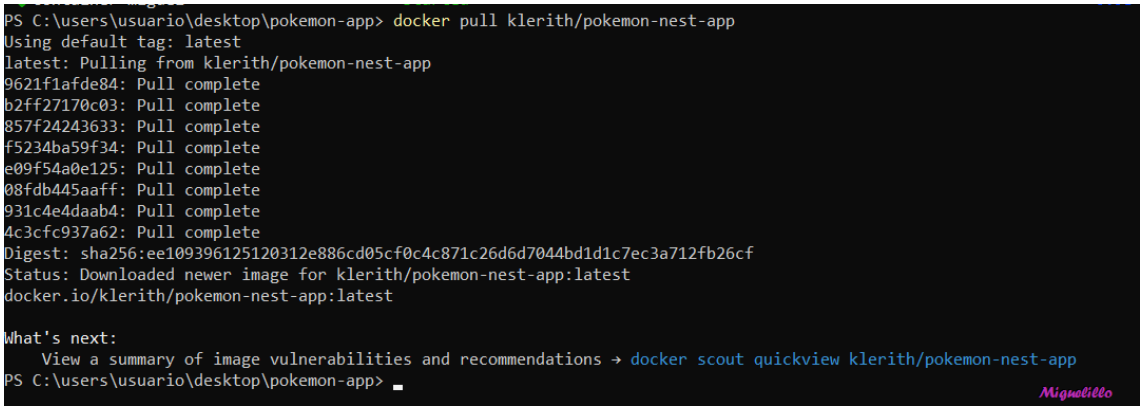
Mongo Express es accesible



34) Descargar la imagen Pokemon-Nest-App

Solución:

Lo descargamos con `docker pull klerith/pokemon-nest-app`



35) Configurar la imagen, el depends, los puertos y always

Solución:

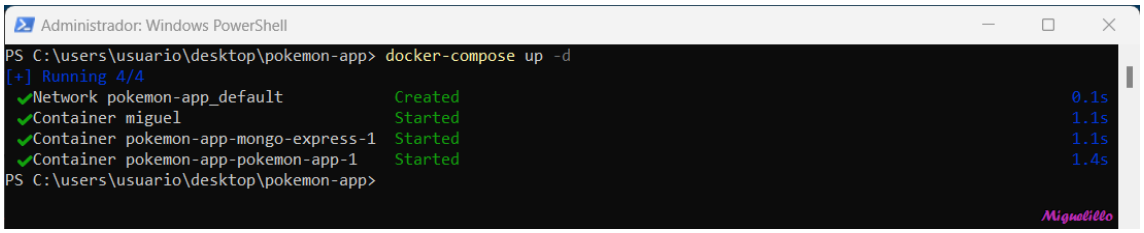
Configuramos todo

```yaml
  pokemon-nest-app:
    container_name: pokemon-nest-app
    image: klerith/pokemon-nest-app
    depends_on:
      - db
      - mongo-express
    ports:
      - "3000:3000"
    restart: always
    environment:
      - MONGODB=mongodb://admin:admin123@miguel:27017

volumes:
  poke-vol:
    external: false
```

36) Probamos el docker-compose

Solución:

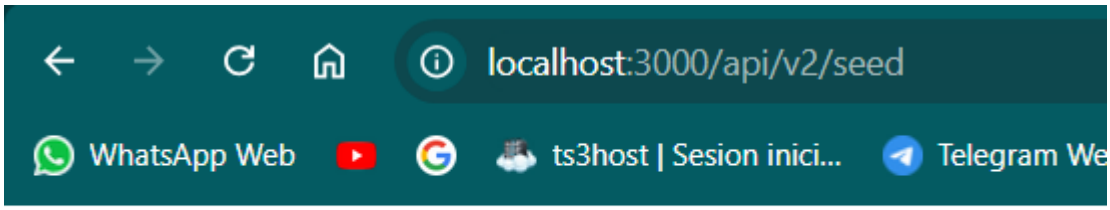
Se crea correctamente



37) Para ejecutar la semilla

Solución:

Ponemos esta URL



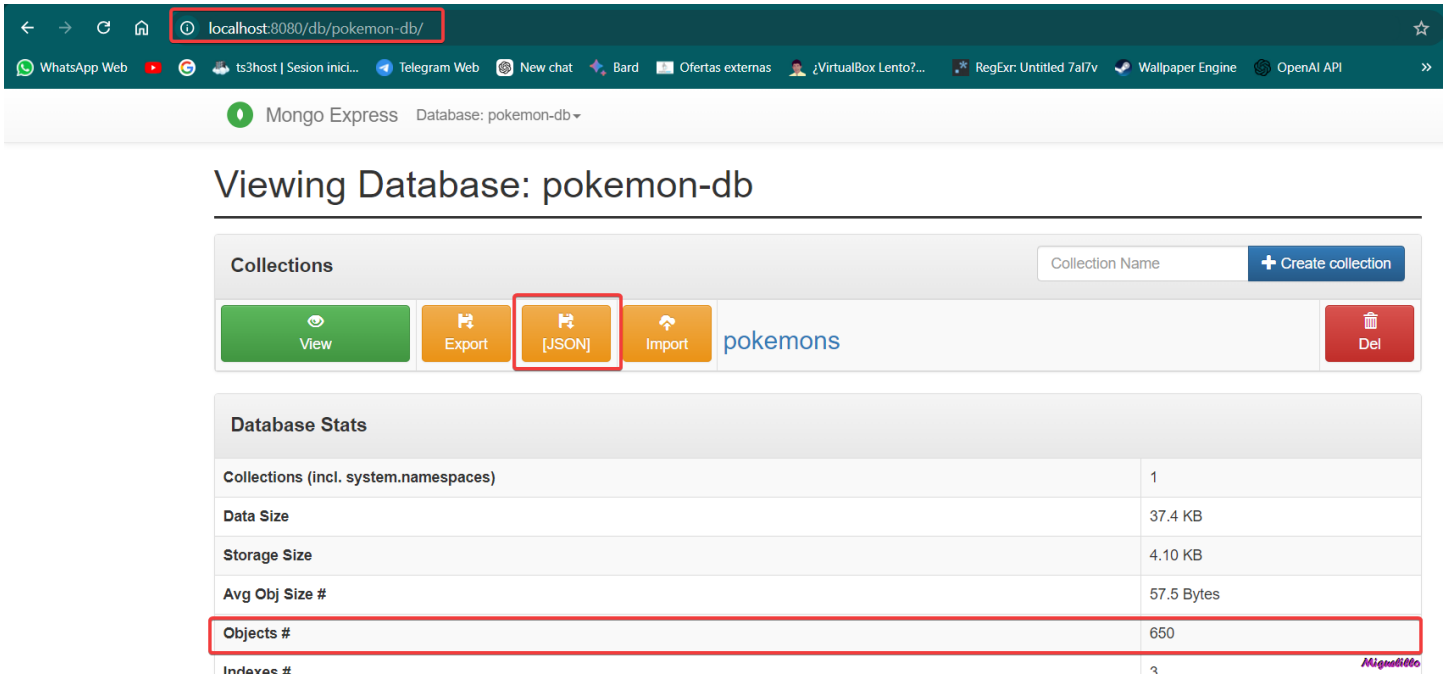
Seed Executed

Miguelillo

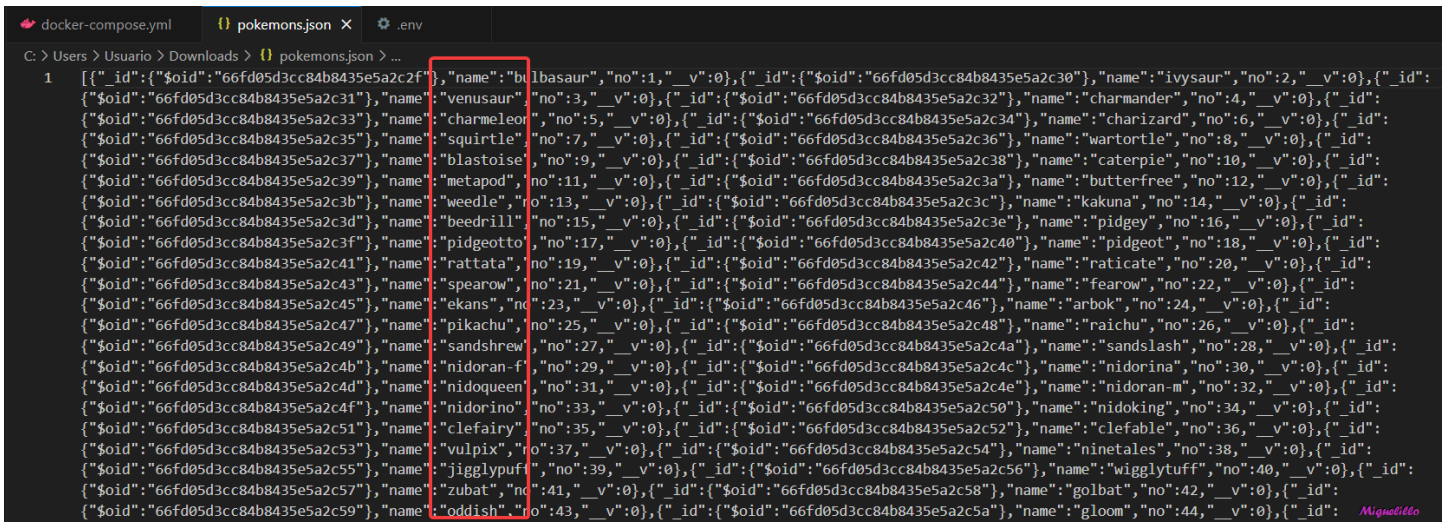
38) Acceder y ver las tablas

Solución:

Vamos al mongo express y descargamos el json de la base de datos, vemos que contiene 650 Objetos



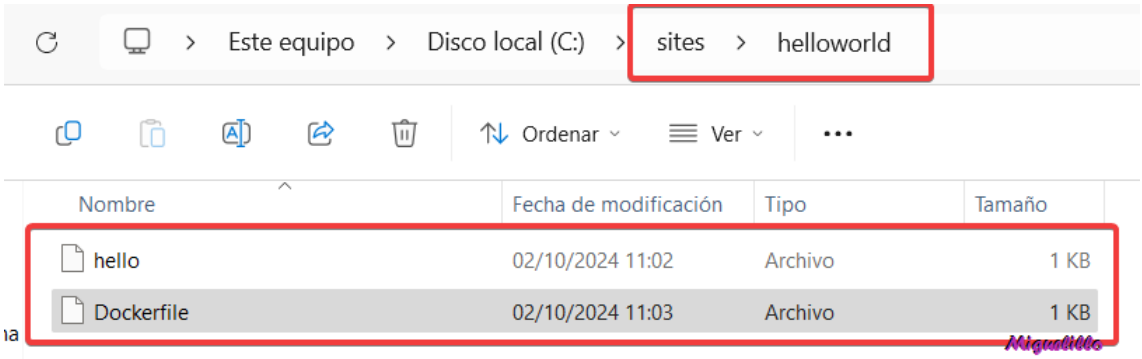
Si abrimos el JSON podemos ver los distintos registros



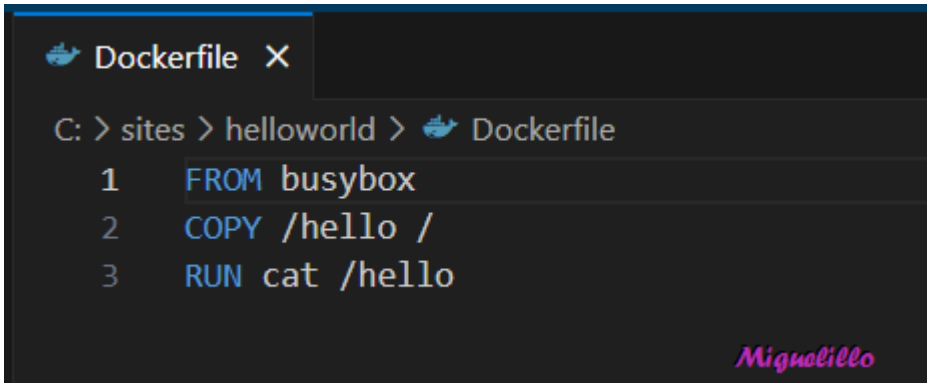
Ejercicio 4

39) Creamos el bulid context y archivos

Solución:



Le añadimos contenido



40) Crearemos la imagen

Solución:

Creamos la imagen con docker build -t helloapp:v1 .

```
PS C:\sites\helloworld> docker build -t helloapp:v1.
ERROR: "docker buildx build" requires exactly 1 argument.
See 'docker buildx build --help'.

Usage:  docker buildx build [OPTIONS] PATH | URL | -

Start a build
PS C:\sites\helloworld> docker build -t helloapp:v1 .
[+] Building 4.5s (9/9) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                0.1s
 => => transferring dockerfile: 80B                                                                                 0.0s
 => [internal] load metadata for docker.io/library/busybox:latest                                                   2.6s
 => [auth] library/busybox:pull token for registry-1.docker.io                                                      0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load build context                                                                                   0.1s
 => => transferring context: 37B                                                                                    0.0s
 => [1/3] FROM docker.io/library/busybox:latest@sha256:768e5c6f5cb6db0794eec98dc7a967f40631746c32232b78a3105fb946  0.8s
 => => resolve docker.io/library/busybox:latest@sha256:768e5c6f5cb6db0794eec98dc7a967f40631746c32232b78a3105fb946  0.0s
 => => sha256:768e5c6f5cb6db0794eec98dc7a967f40631746c32232b78a3105fb946f3ab83 10.20kB / 10.20kB                    0.0s
 => => sha256:22f27168517de1f58dae0ad51eacf1527e7e7ccc47512d3946f56bdbe913f564 610B / 610B                          0.0s
 => => sha256:27a71e19c95622dce4d60d4a3760707495c9875f5c5322c5bd535214799593ce 372B / 372B                          0.0s
 => => sha256:a46fbb00284bdd1a1d8d80d51333abc851371a7b8d44cc781c4469b5e54119ae 2.16MB / 2.16MB                      0.4s
 => => extracting sha256:a46fbb00284bdd1a1d8d80d51333abc851371a7b8d44cc781c4469b5e54119ae                           0.2s
 => [2/3] COPY /hello /                                                                                             0.1s
 => [3/3] RUN cat /hello                                                                                            0.5s
 => exporting to image                                                                                              0.1s
 => => exporting layers                                                                                             0.1s
 => => writing image sha256:5f2cd69b52ca827ba33d1802bf804db4debe110ad2bf5908818f6c2c73ff9701                        0.0s
 => => naming to docker.io/library/helloapp:v1                                                                      0.0s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\sites\helloworld>
```

Vemos las imágenes instalada

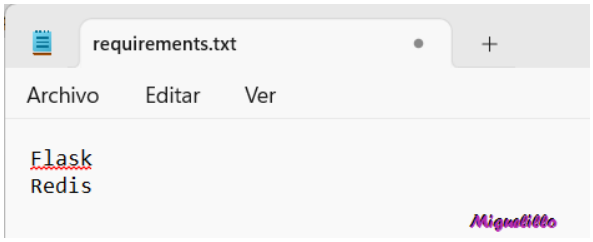
```
Administrador: Windows PowerShell
PS C:\sites\helloworld> docker images
REPOSITORY                TAG       IMAGE ID       CREATED              SIZE
helloapp                  v1        5f2cd69b52ca   About a minute ago   4.27MB
mongo                     6.0       2835e1541e70   5 weeks ago          716MB
postgres                  latest    b781f3a53e61   7 weeks ago          432MB
httpd                     latest    9cb0a2315602   2 months ago         148MB
mongo-express             latest    870141b735e7   6 months ago         182MB
hello-world               latest    d2c94e258dcb   17 months ago        13.3kB
postgres                  15.1      ccd94e8b5fd9   20 months ago        379MB
klerith/pokemon-nest-app  latest    60b65d5c4cad   21 months ago        402MB
docker/getting-started    latest    3e4394f6b72f   21 months ago        47MB
dpage/pgadmin4            6.17      7ac3e08c2412   22 months ago        358MB
PS C:\sites\helloworld>
```

41) Creando aplicaciones en contenedores

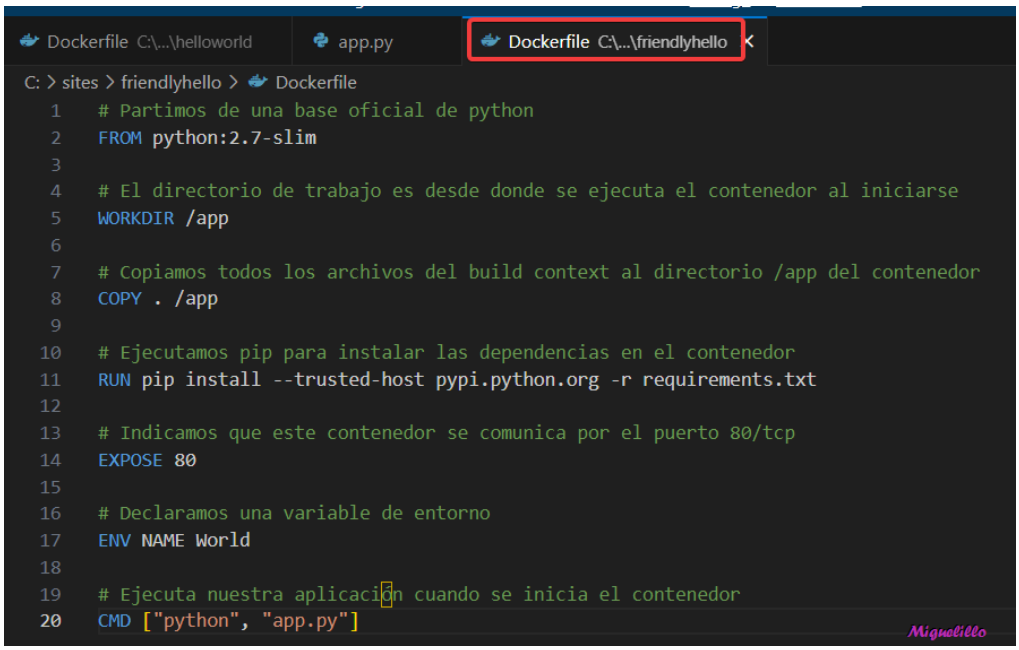
Solución:

Creamos la carpeta friendlyhello y dentro cremamos app.py, le ponemos el siguiente contenido

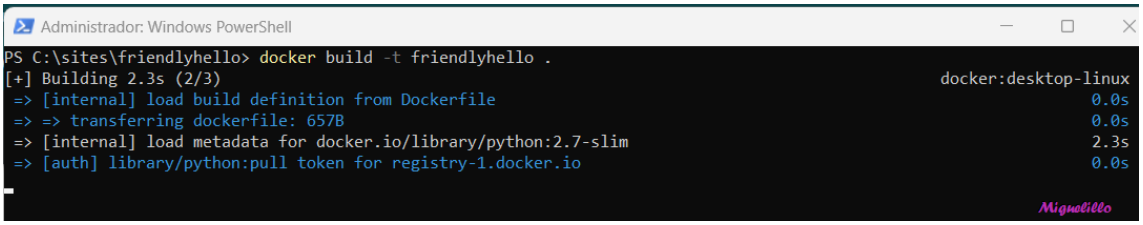
Creamos el archivo **requeriments.txt**



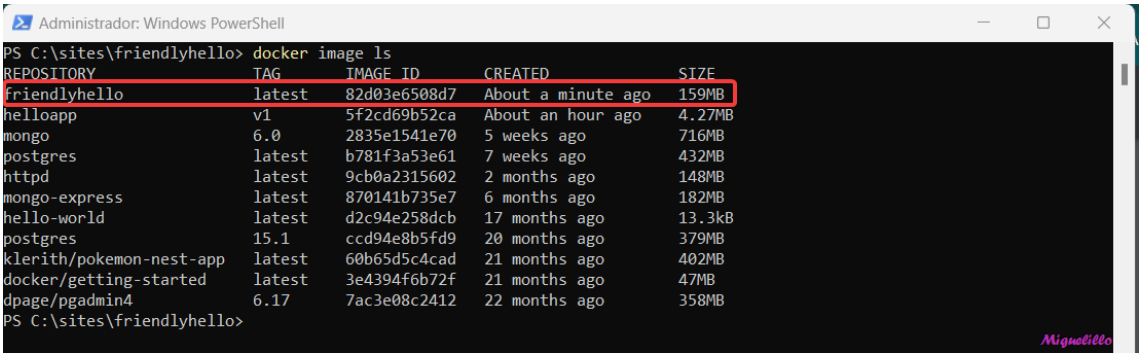
Creamos el archivo **Dockerfile**



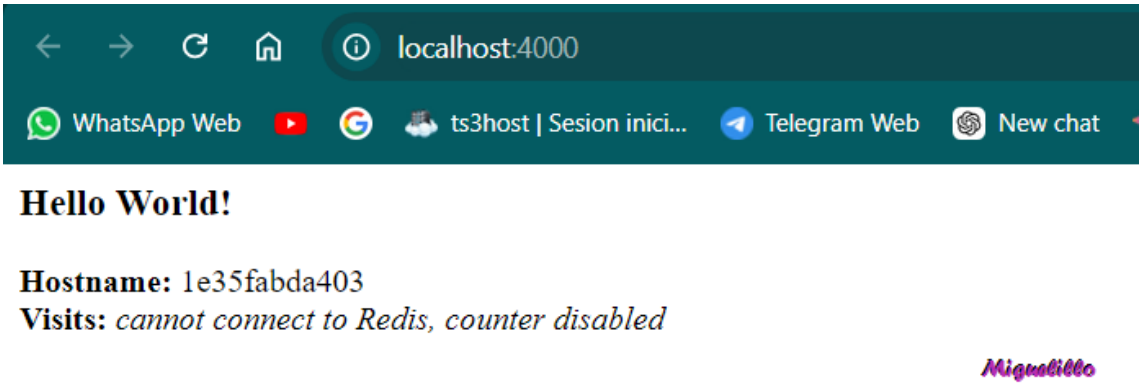
Construimos la imagen de nuestra aplicación



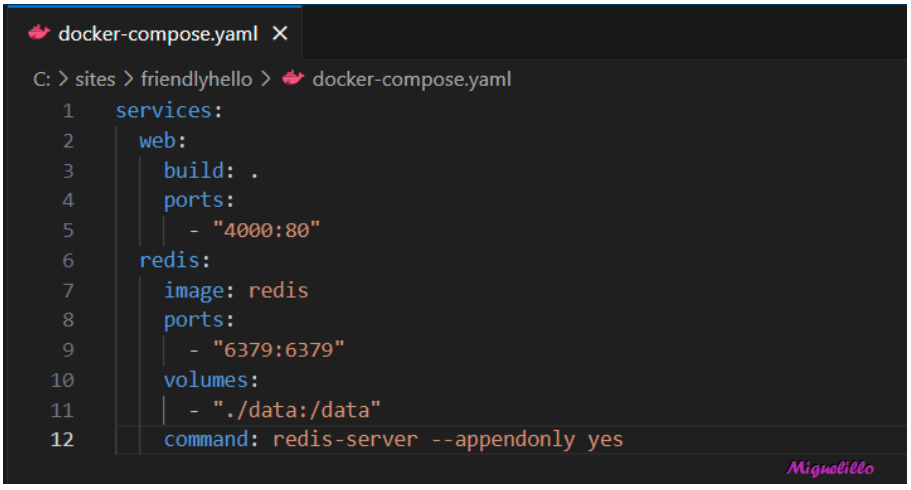
Comprobamos que se ha creado



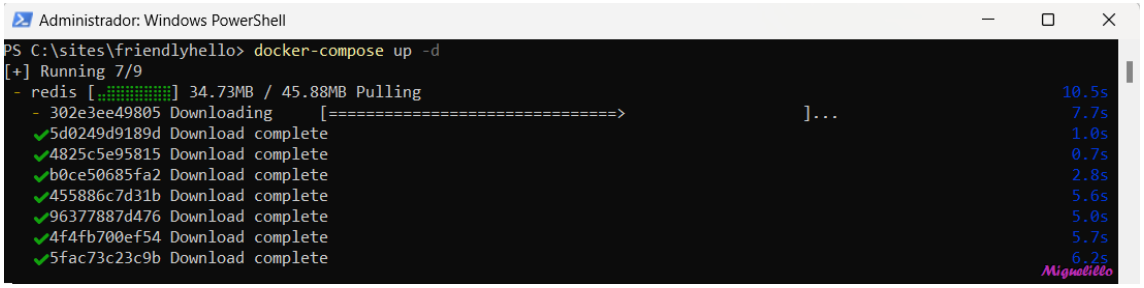
Probamos la imagen corre correctamente



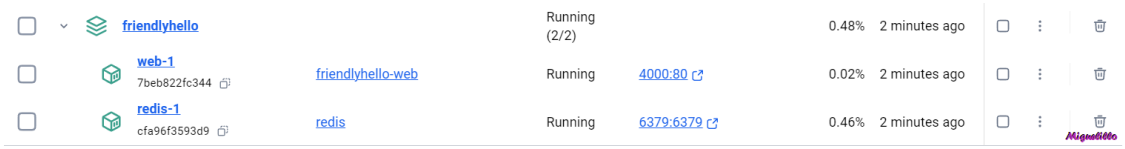
Hacemos un docker-compose



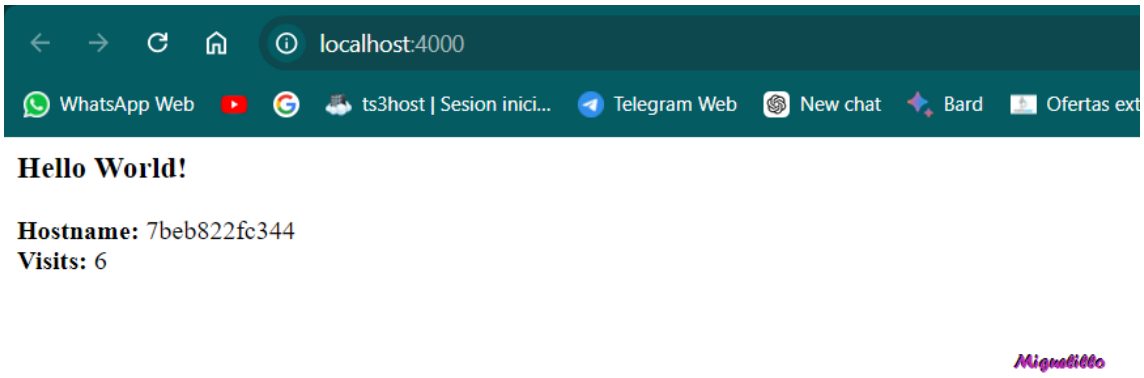
Vemos como se crea



Vemos como se ha creado



Lo probamos



42) Balanceo de carga

Solución:

Actualizamos el yaml anterior, cambiando el ports por un lb (un balanceador de carga)

docker-compose.yaml

```
C: > sites > friendlyhello > docker-compose.yaml
1   services:
2     web:
3       build: .
4     redis:
5       image: redis
6       volumes:
7         - "./data:/data"
8       command: redis-server --appendonly yes
9     lb:
10      image: dockercloud/haproxy
11      ports:
12        - 4000:80
13      links:
14        - web
15      volumes:
16        - /var/run/docker.sock:/var/run/docker.sock
```

Arrancamos la aplicación

```
PS C:\sites\friendlyhello> docker-compose up -d --scale web=5
[+] Running 2/4
 - lb [⣿⣿⣿⣿] 6.399MB / 16.38MB Pulling                                    10.5s
   ✔1160f4abea84 Pull complete                                              3.8s
   ✔b0df9c632afc Pull complete                                              3.9s
   - a49b18c7cd3a Downloading [==============>                    ]  3.98...   7.1s
```

Esperamos a que se inicien todos los servicios

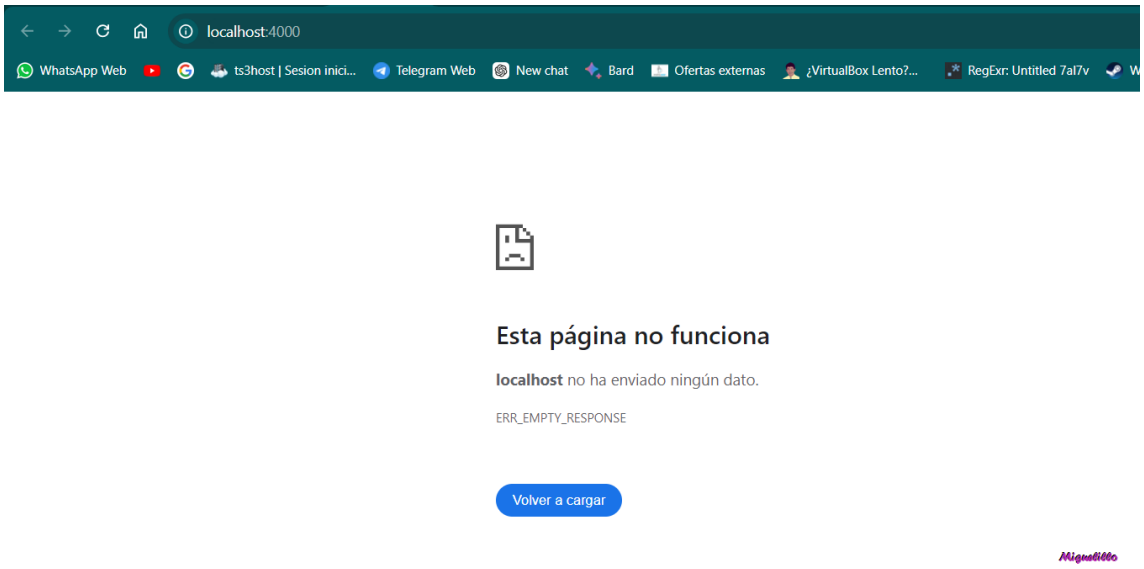
```
PS C:\sites\friendlyhello> docker-compose up -d --scale web=5
[+] Running 4/4
 ✔lb Pulled                                                                 21.0s
   ✔1160f4abea84 Pull complete                                              3.8s
   ✔b0df9c632afc Pull complete                                              3.9s
   ✔a49b18c7cd3a Pull complete                                             17.6s
[+] Running 7/7
 ✔Container friendlyhello-redis-1  Started                                   1.0s
 ✔Container friendlyhello-web-5    Started                                   2.8s
 ✔Container friendlyhello-web-2    Started                                   1.6s
 ✔Container friendlyhello-web-1    Started                                   1.2s
 ✔Container friendlyhello-web-4    Started                                   2.1s
 ✔Container friendlyhello-web-3    Started                                   2.5s
 ✔Container friendlyhello-lb-1     Started                                   2.8s
PS C:\sites\friendlyhello>
```

Vemos que se inician todos los servicios

Administrador: Windows PowerShell

```
PS C:\sites\friendlyhello> docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED         STATUS         PORTS
              NAMES
3f9432516979   dockercloud/haproxy   "/sbin/tini -- docke…"   2 minutes ago   Up 2 minutes   443/tcp, 1936/tcp, 0.0.0.0:
4000->80/tcp   friendlyhello-lb-1
ddd65f20aafc   friendlyhello-web     "python app.py"          2 minutes ago   Up 2 minutes   80/tcp
              friendlyhello-web-1
444e2ee076e8   friendlyhello-web     "python app.py"          2 minutes ago   Up 2 minutes   80/tcp
              friendlyhello-web-2
48805059bf0a   friendlyhello-web     "python app.py"          2 minutes ago   Up 2 minutes   80/tcp
              friendlyhello-web-4
f1e36d82268b   friendlyhello-web     "python app.py"          2 minutes ago   Up 2 minutes   80/tcp
              friendlyhello-web-3
fce35aa8281b   redis                 "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   6379/tcp
              friendlyhello-redis-1
5da01b6c695b   friendlyhello-web     "python app.py"          2 minutes ago   Up 2 minutes   80/tcp
              friendlyhello-web-5
PS C:\sites\friendlyhello>
```

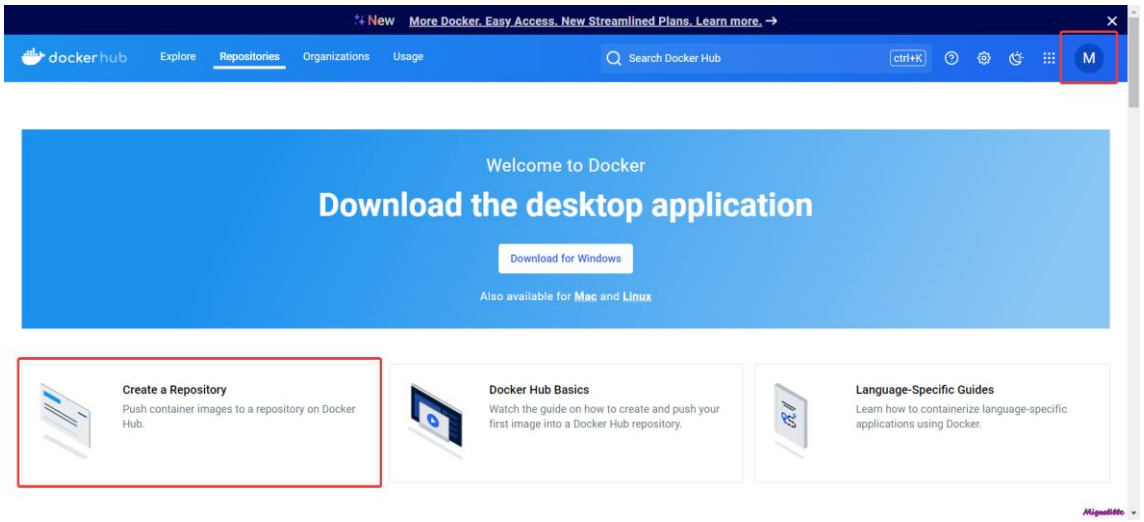
No funciona tiene un fallo



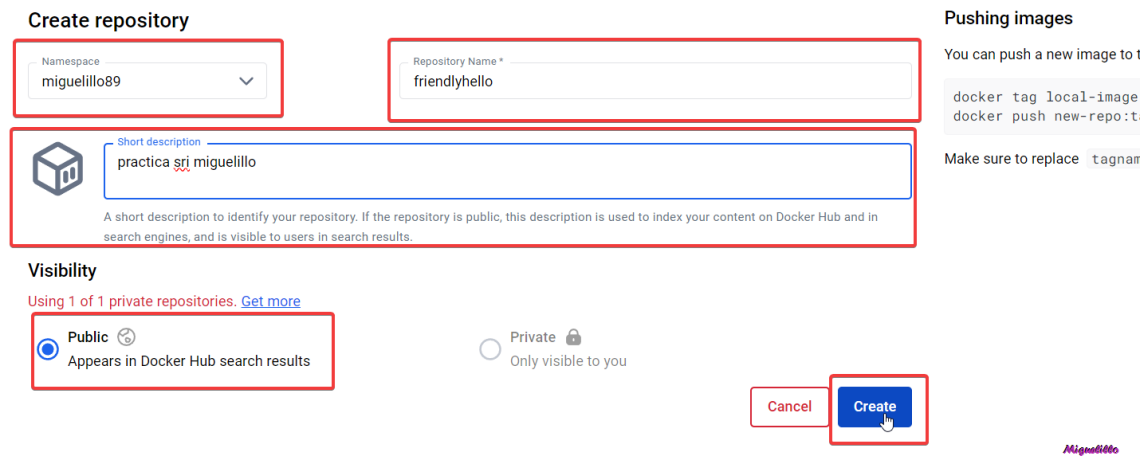
43) Compartir imágenes

Solución:

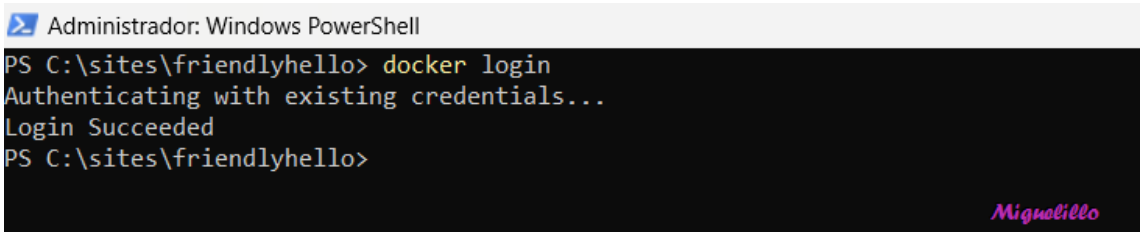
Nos creamos una cuenta y clicamos en **Create a Repository**



Le ponemos un nombre y descripción

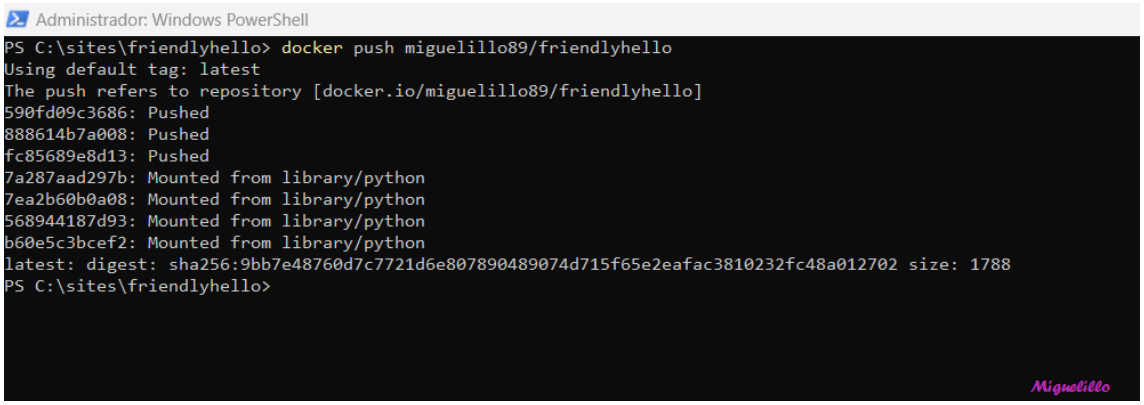


Iniciamos sesión en dockers

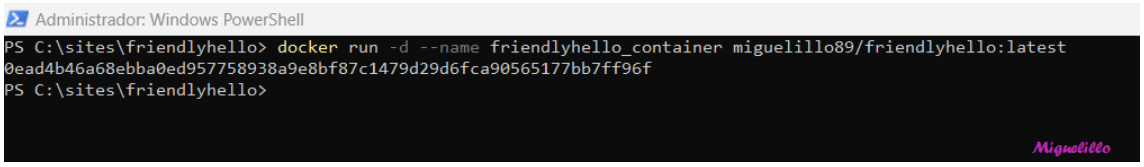


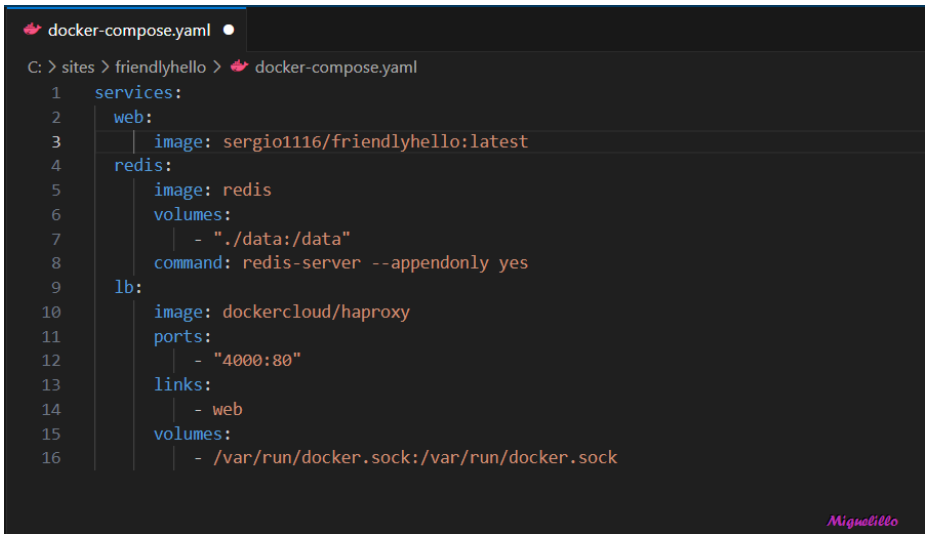
Subimos el dockerfile

Lo subimos



Probamos a usar la imagen





docker-compose.yaml

C: > sites > friendlyhello > docker-compose.yaml

```yaml
services:
  web:
    image: miguelillo89/friendlyhello:latest
  redis:
    image: redis
    volumes:
      - "./data:/data"
    command: redis-server --appendonly yes
  lb:
    image: dockercloud/haproxy
    ports:
      - "4000:80"
    links:
      - web
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
```

Miguelillo

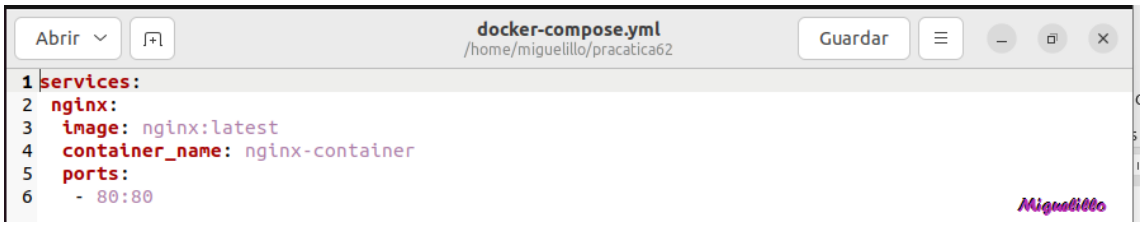
Ejercicio 5

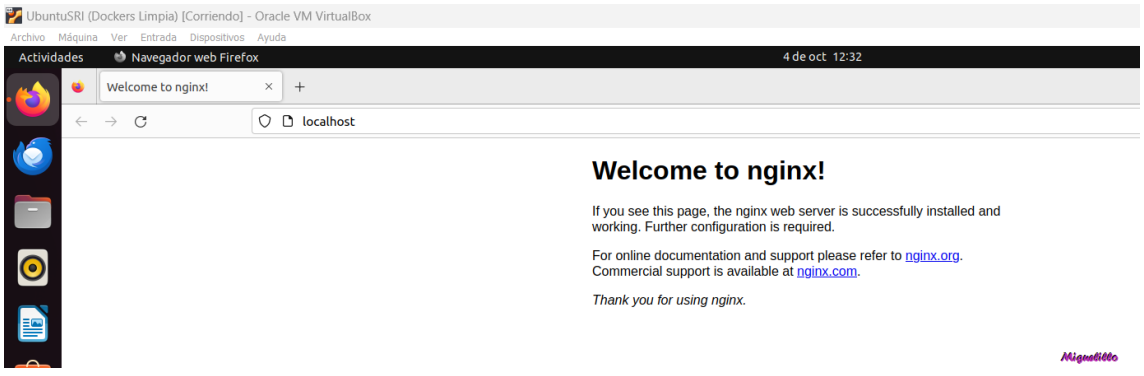
44) Instalamos NGINX

Solución:

Creamos la estructura de directorios para el ejercicio

```
miguelillo@miguelilloSRI:~/pracatica62$ ls -r
www  php  nginx  docker-compose.yml
miguelillo@miguelilloSRI:~/pracatica62$ ls www/
html
miguelillo@miguelilloSRI:~/pracatica62$ ls www/html/
index.php
miguelillo@miguelilloSRI:~/pracatica62$ ls php/
Dockerfile
miguelillo@miguelilloSRI:~/pracatica62$ ls nginx/
default.conf  Dockerfile
```

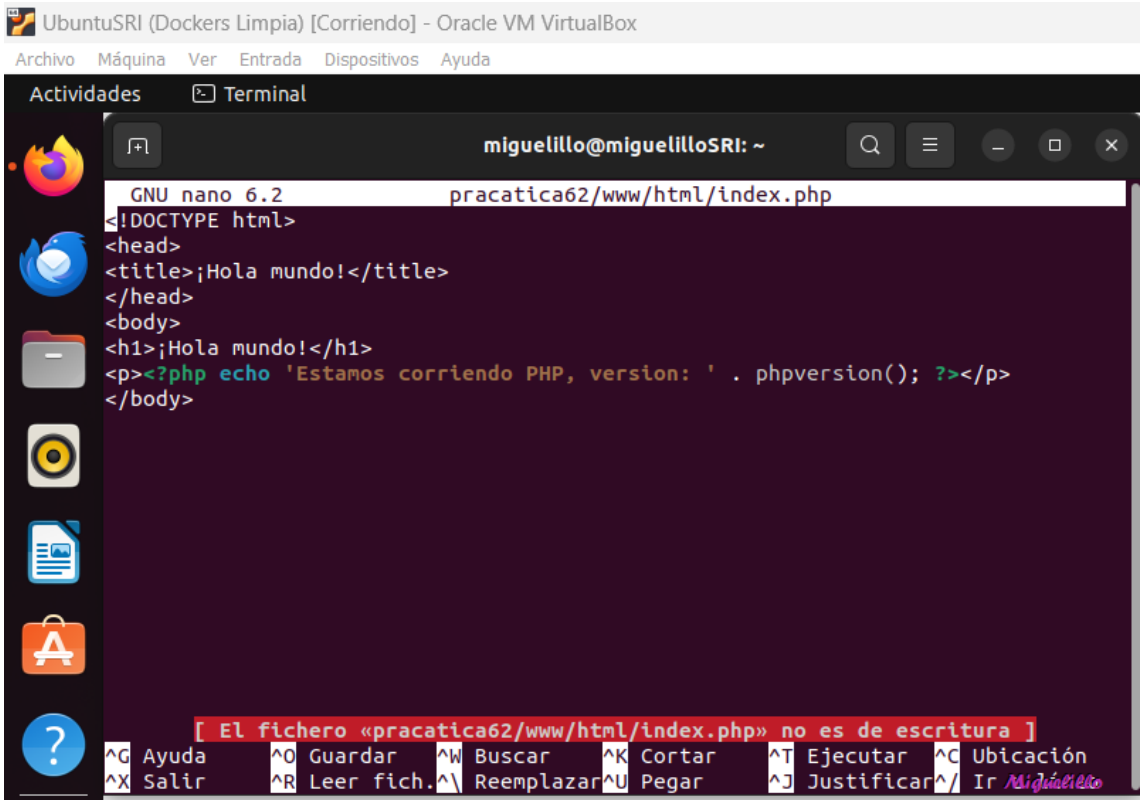





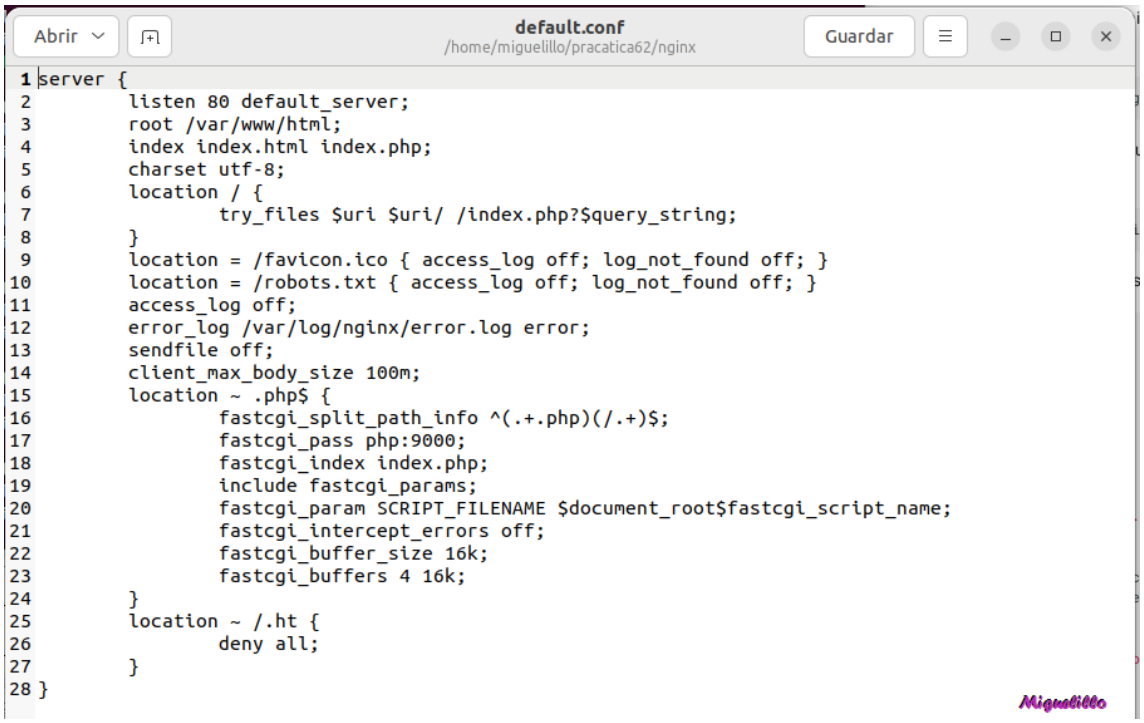
45) Creación contenedor PHP

Solución:

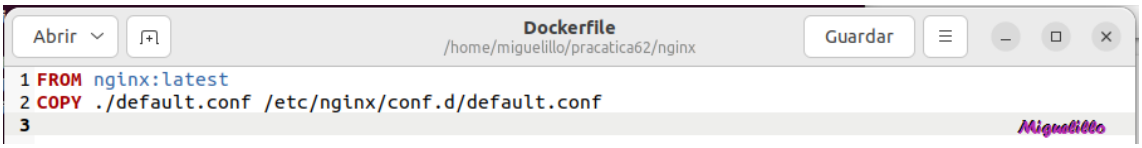
Dentro de index.php añadimos el siguiente código



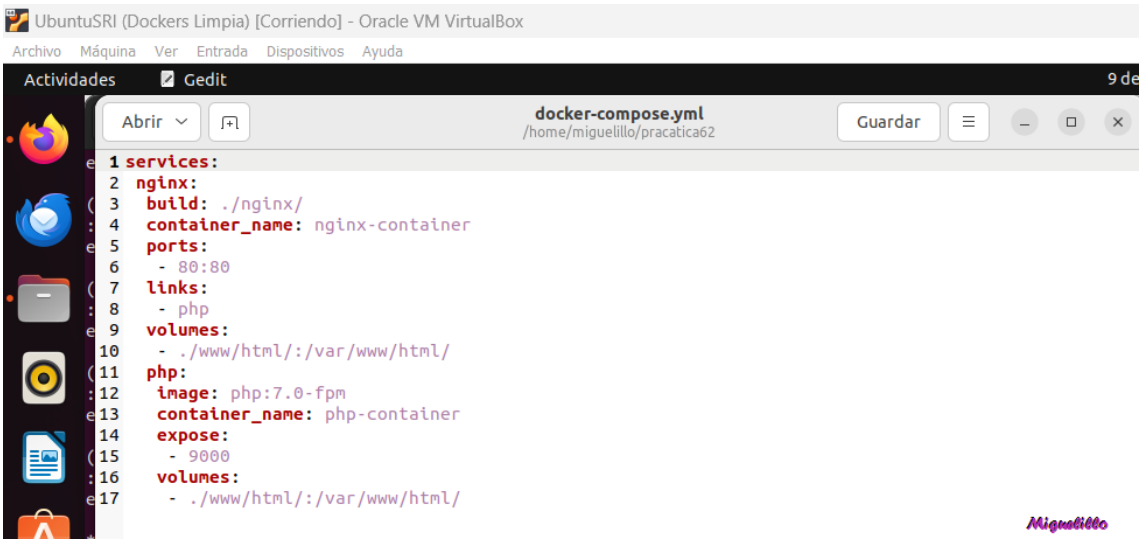
Modificamos el default.conf



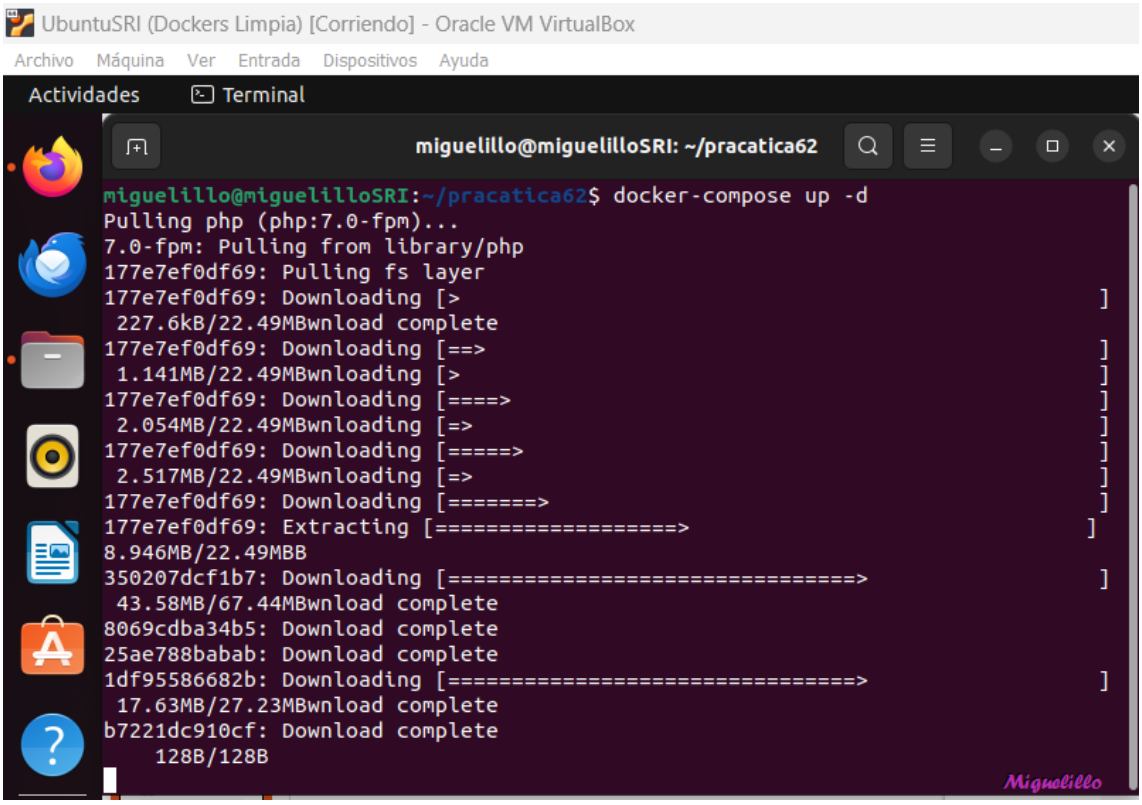
Añadimos el contenido al dockerfile



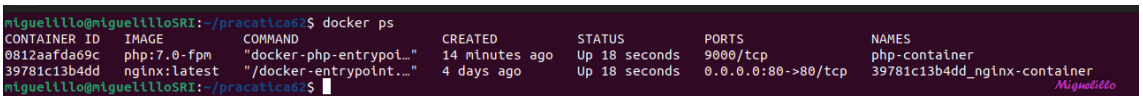
Modificamos el docker-compose



Lo ejecutamos



Se ha creado y está corriendo



46) Creación de un contenedor para datos

Solución:

Modificamos el docker-compose

Abrir — docker-compose.yml — /home/miguelillo/pracatica62 — Guardar

```
services:
 nginx:
  build: ./nginx/
  container_name: nginx-container
  ports:
   - 80:80
  links:
   - php
  volumes:
   - ./www/html/:/var/www/html/
 php:
  image: php:7.0-fpm
  container_name: php-container
  expose:
   - 9000
  volumes:
   - ./www/html/:/var/www/html/
 app-data:
  image: php:7.0-fpm
  container_name: app-data-container
  volumes:
   - ./www/html/:/var/www/html/
  command: "true"
```

Miguelillo

Probamos que se cree

dades Terminal — 9 de oct 11:54 — miguelillo@miguelilloSRI: ~/pracatica62

```
miguelillo@miguelilloSRI:~/pracatica62$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                      PORTS                  NAMES
0812aafda69c   php:7.0-fpm    "docker-php-entrypoi…"   56 minutes ago   Up 42 minutes               9000/tcp               php-container
39781c13b4dd   nginx:latest   "/docker-entrypoint.…"   4 days ago       Up 42 minutes               0.0.0.0:80->80/tcp     39781c13b4dd_nginx-container
fc0c0f6bc484   httpd          "httpd-foreground"       5 days ago       Exited (0) 4 days ago                              miguel
ccf20e8165a0   hello-world    "/hello"                 5 days ago       Exited (0) 5 days ago                              interesting_fermi
miguelillo@miguelilloSRI:~/pracatica62$
```

Miguelillo

UbuntuSRI (Dockers Limpia) [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Actividades — Navegador web Firefox

¡Hola mundo! — Nueva pestaña — Problema al cargar la pág — Install t

localhost

¡Hola mundo!

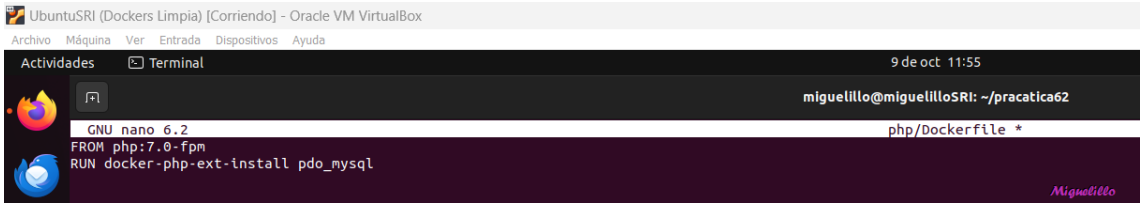
Estamos corriendo PHP, version: 7.0.33

Miguelillo

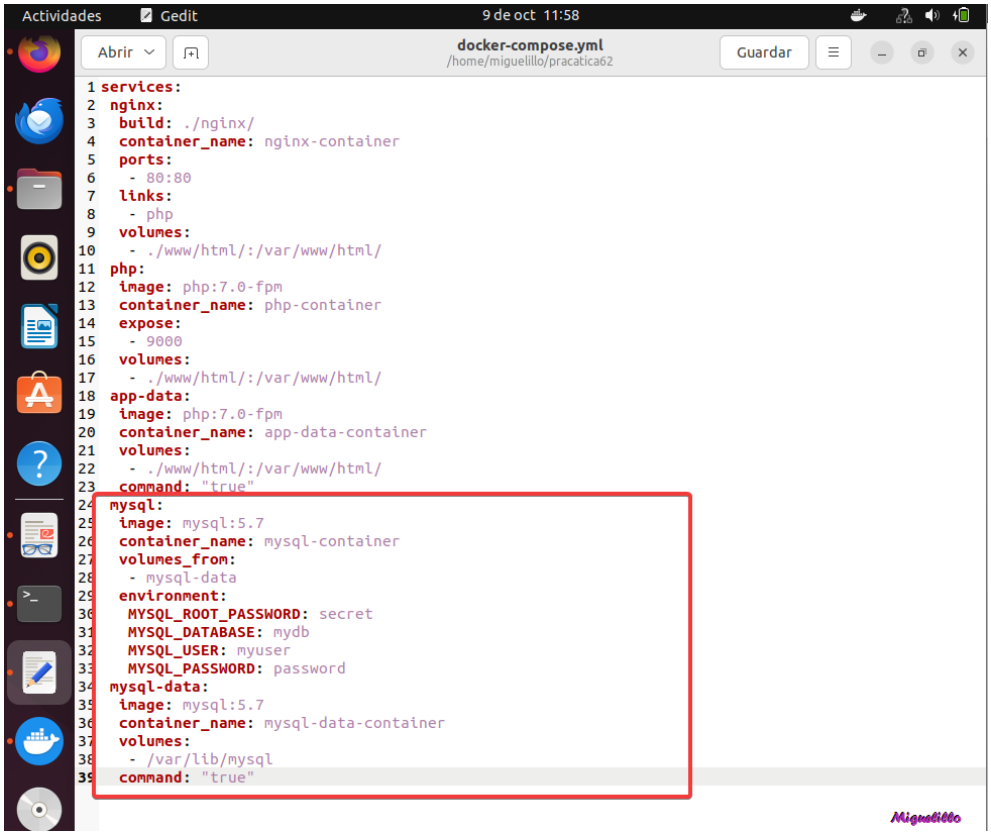
47) Creación de contenedor MySQL

Solución:

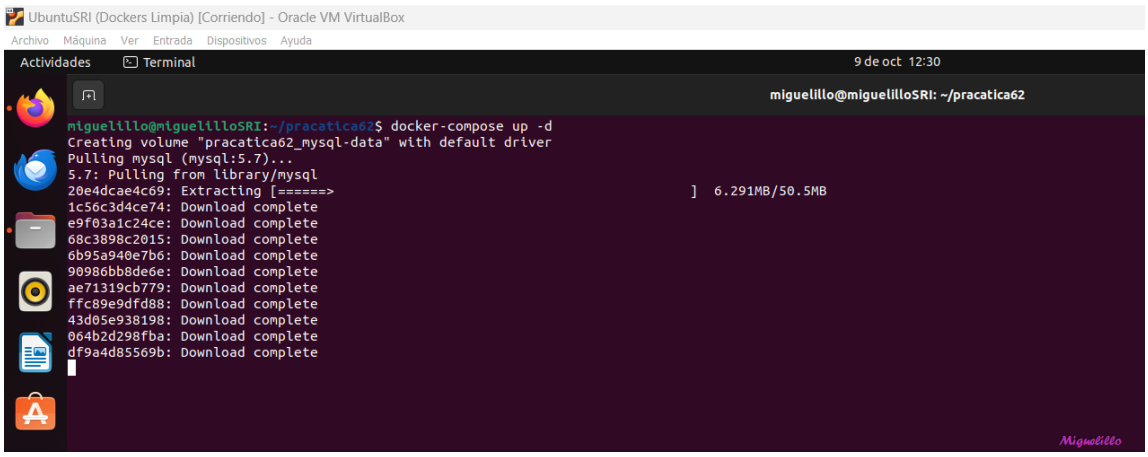
Añadimos información al Dockerfile



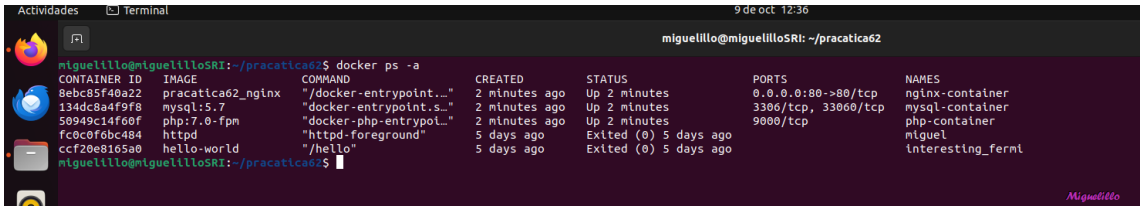
Modificamos el docker compose



Lo ejecutamos



Probamso que se ejecute



Probamos a acceder a la web funciona correctamente