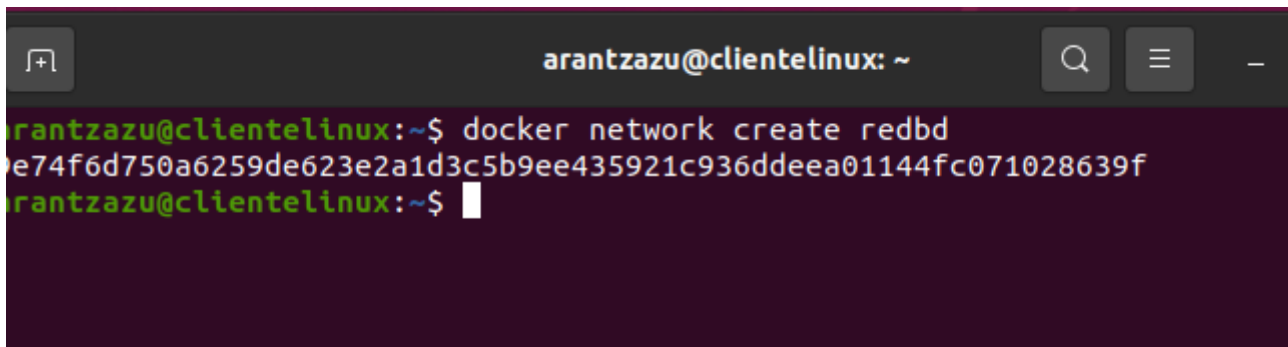


Ejercicio 3 - Redes

Creamos una red bridge, llamada redbd

```
docker network create redbd
```



```
arantzazu@clientlinux: ~  
arantzazu@clientlinux:~$ docker network create redbd  
e74f6d750a6259de623e2a1d3c5b9ee435921c936ddeea01144fc071028639f  
arantzazu@clientlinux:~$
```

como no hemos indicado ninguna configuración en la red que hemos creado, docker asigna un direccionamiento a la red:

```
docker network inspect redbd
```

```

arantzazu@clientlinux:~$ docker network inspect redbd
[
  {
    "Name": "redbd",
    "Id": "9e74f6d750a6259de623e2a1d3c5b9ee435921c936ddeea01144fc071028639f",
    "Created": "2022-03-21T18:02:44.591312445+01:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.19.0.0/16",
          "Gateway": "172.19.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]
arantzazu@clientlinux:~$

```

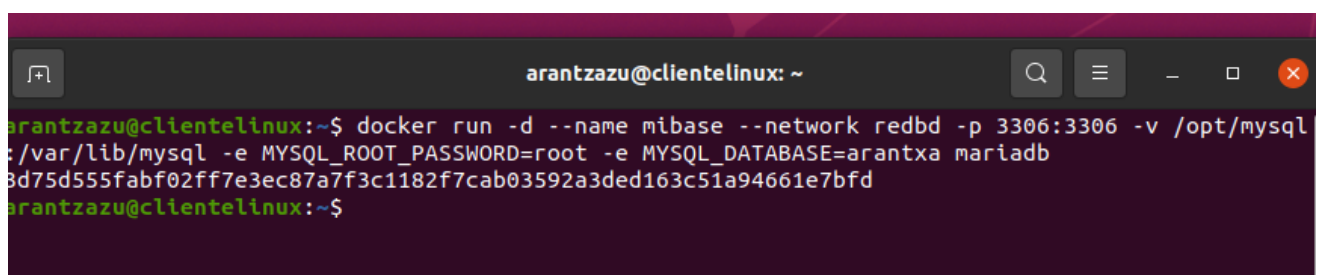
Ahora creamos un contenedor con una imagen de mariaDB que estará en la red **redbd**.

Se ejecutará en segundo plano (-d) y será accesible desde el puerto 3306. Con el usuario root con contraseña root y con un volumen de datos persistente (-v).

```

docker run -d --name mibase --network redbd -p 3306:3306 -v
/opt/mysql:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e
MYSQL_DATABASE=arantxa mariadb

```



```

arantzazu@clientlinux: ~
arantzazu@clientlinux:~$ docker run -d --name mibase --network redbd -p 3306:3306 -v /opt/mysql
:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=arantxa mariadb
3d75d555fabf02ff7e3ec87a7f3c1182f7cab03592a3ded163c51a94661e7bfd
arantzazu@clientlinux:~$

```

Ahora creamos otro contenedor con una imagen Adminer , que se pueda conectar al contenedor **mibase** (con la imagen mariadb) anteriormente creado (lo hago a través de --link). Los dos contenedores tienen que estar en la misma red (--network redbd). El puerto no he puesto el 8080, como pone la documentacion de la imagen de Adminer de Docker Hub, sino el 8083, porque por el puerto 8080, me daba un error que no pude subsanar por mucho que lo he intentado.

```
docker run --name adminer-ari --link mi_base:db --network redbd -p 8083:8080 -d adminer
```

```
arantzazu@clientlinux: ~
arantzazu@clientlinux:~$ docker run --name adminer-ari --link mi_base:db --network redbd -p 8083:8080 -d adminer
Unable to find image 'adminer:latest' locally
latest: Pulling from library/adminer
3d2430473443: Pull complete
459cf206553a: Pull complete
672bda396649: Pull complete
71751fed054d: Pull complete
65307cbea578: Pull complete
03d388f8715a: Pull complete
effa5850281f: Pull complete
d24018452f30: Pull complete
cb89724932bf: Pull complete
a85997cb903e: Pull complete
6deaf2338558: Pull complete
4c2f9193e2df: Pull complete
1dd22076027b: Pull complete
420eba3cf3ed: Pull complete
fb2f68eb3f87: Pull complete
Digest: sha256:1c01d3eef3bc9e60a43571dafb48d459c32faa4a1ec4d6c1fa0a9a172071999f
Status: Downloaded newer image for adminer:latest
8db73b54712ebb5620ec1ec19dfc07a2d03cf3ec2d8996032ee48a8ee1e8a812
arantzazu@clientlinux:~$
```

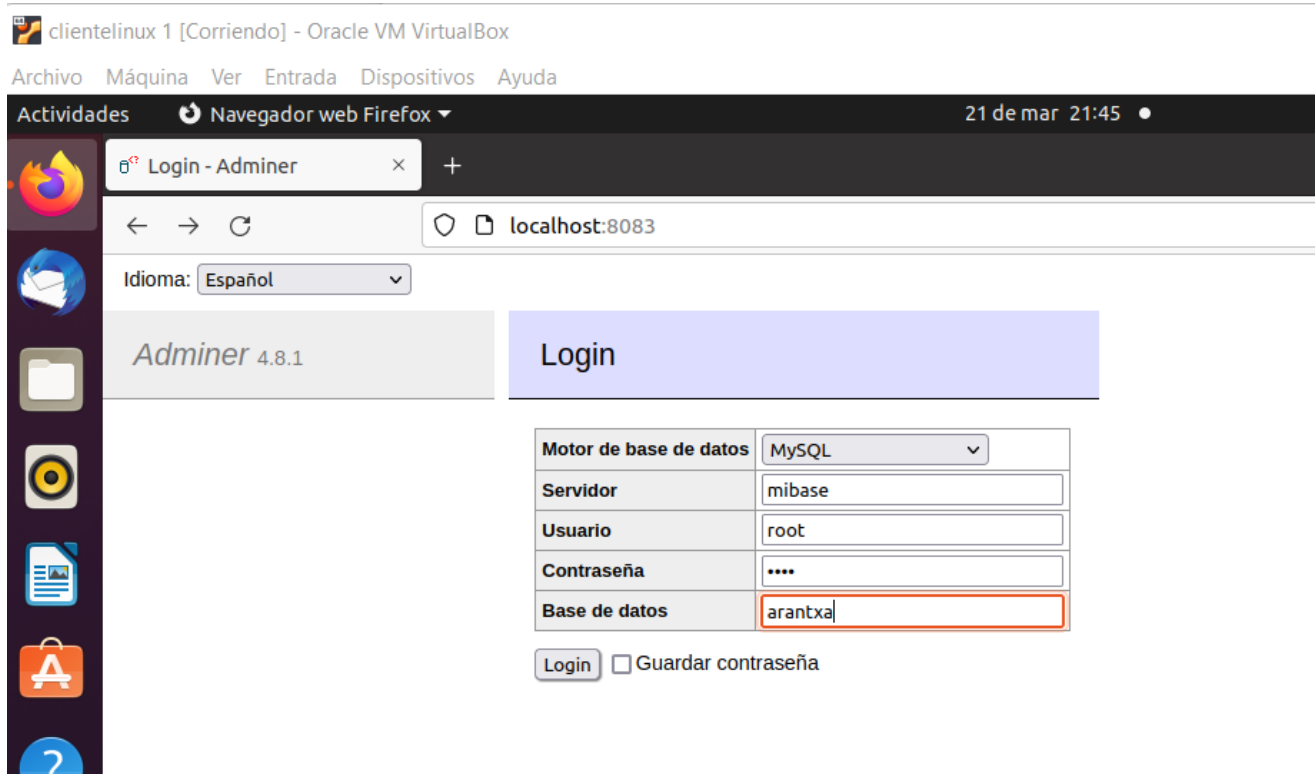
Captura de los dos contenedores creados

```
docker ps
```

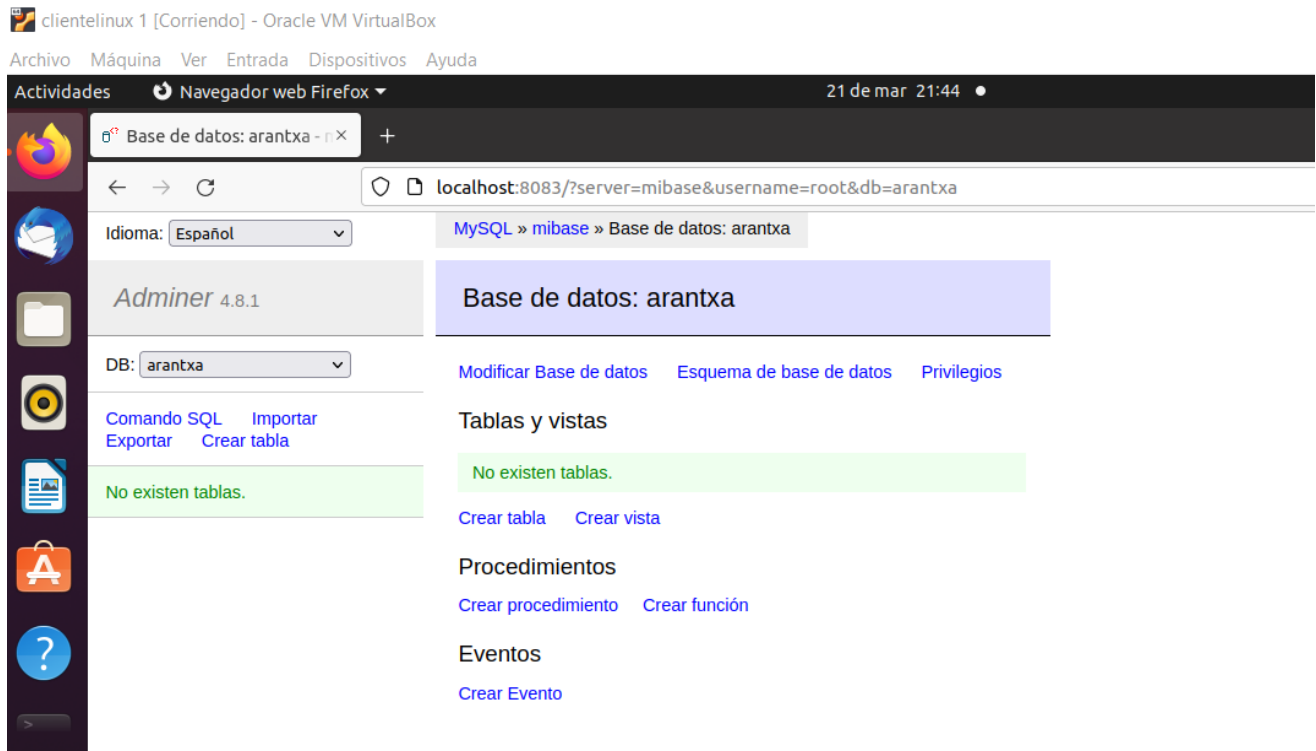
```
arantzazu@clientlinux: ~
arantzazu@clientlinux:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
db73b54712e   adminer    "entrypoint.sh docke..." About a minute ago Up About a minute   0.0.0.0:8083->8080/tcp, :::8083->8080/tcp   adminer-ari
d75d555fabf   mariadb    "docker-entrypoint.s..." 2 hours ago   Up 2 hours     0.0.0.0:3306->3306/tcp, :::3306->3306/tcp   mibase
```

Ahora abrimos una ventana del navegador firefox, accedo a :

<http://localhost:8083> y podemos ver el interface de Adminer 4.8.1

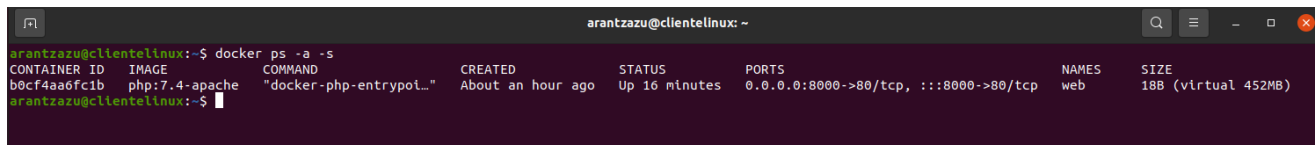


Captura de la interfaz de Adminer, donde se puede ver la base de datos creada (base de datos: arantxa) :



Accedo al contenedor creado con mariadb y llamado **mibase**

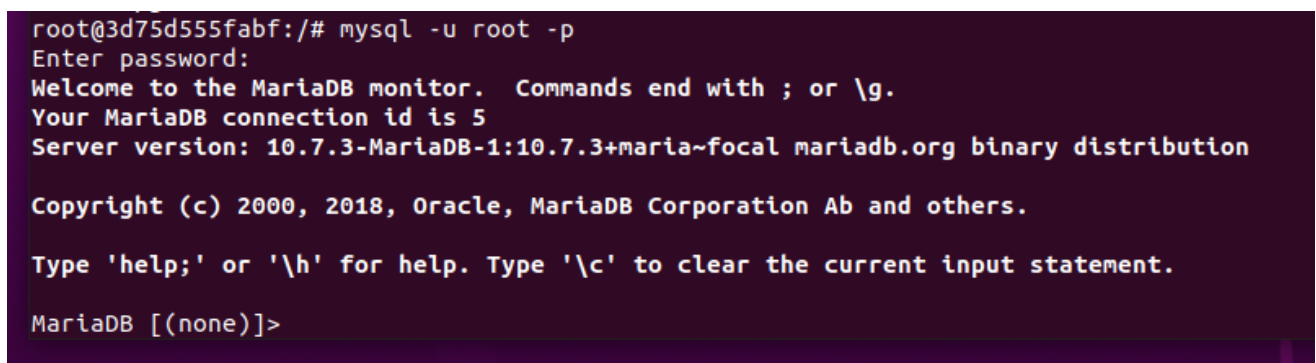
```
docker exec -it mibase bash
```



Accedo a la base de datos (con usuario root)

```
mysql -u root -p
```

Nos pide contraseña:



Ahora vemos la base de datos creada:

```
show databases;
```

```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| arantxa  |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.001 sec)

MariaDB [(none)]> 
```

Salimos del contenedor:

```
exit
```

```
exit
```

```
MariaDB [(none)]> exit
Bye
root@3d75d555fabf:/# exit
exit
arantzazu@clientlinux:~$ 
```

Ahora detenemos y eliminamos los contenedores:

```
docker stop $(docker ps -a -q)
```

```
arantzazu@clientlinux:~$ docker stop $(docker ps -a -q)
8db73b54712e
3d75d555fabf
arantzazu@clientlinux:~$
```

```
docker rm $(docker ps -a -q)
```

```
arantzazu@clientlinux:~$ docker rm $(docker ps -a -q)
8db73b54712e
3d75d555fabf
arantzazu@clientlinux:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
arantzazu@clientlinux:~$
```

(* con `docker ps -a` , comprobamos que ya no hay contenedores)

Ahora enumeramos los volúmenes creados:

```
docker volume ls
```

```
arantzazu@clientelinux: ~  
arantzazu@clientelinux:~$ docker volume ls  
DRIVER      VOLUME NAME  
local       8e76941ae363db8e869182b7801f069bb011085540e5d177dfbf328dab9323d8
```

Y lo eliminamos (con este comando eliminamos todos los volúmenes existentes, yo tenía otro creado de prueba llamado `mi_volumen` y con esto lo eliminará también):

```
docker volume prune
```

```
arantzazu@clientelinux:~$ docker volume prune  
WARNING! This will remove all local volumes not used by at least one container.  
Are you sure you want to continue? [y/N] y  
Deleted Volumes:  
8e76941ae363db8e869182b7801f069bb011085540e5d177dfbf328dab9323d8  
mi_volumen  
  
Total reclaimed space: 143.2MB  
arantzazu@clientelinux:~$
```

Eliminamos la red creada, primero la visualizo (la red creada se llamaba **redbd**):

```
docker network ls
```

```
arantzazu@clientelinux: ~  
arantzazu@clientelinux:~$ docker network ls  
NETWORK ID      NAME      DRIVER      SCOPE  
339ee845b6ba    bridge    bridge      local  
57b603e8e2cd    host      host        local  
9fa4a2616fbc    none      null        local  
9e74f6d750a6    redbd     bridge      local  
arantzazu@clientelinux:~$
```

Y ahora la borro:

```
docker network rm 9e74f6d750a6
```

```
arantzazu@clientelinux:~$ docker network rm 9e74f6d750a6  
9e74f6d750a6  
arantzazu@clientelinux:~$ docker network ls  
NETWORK ID      NAME      DRIVER      SCOPE  
339ee845b6ba    bridge    bridge      local  
57b603e8e2cd    host      host        local  
9fa4a2616fbc    none      null        local  
arantzazu@clientelinux:~$
```

webgrafía

Codec : <https://www.youtube.com/watch?v=4.9VKzKBtbNY>

Digital Ocean: <https://digitalocean.com>