DOCKER Y SUBNETING

16 DE DICIEMBRE DE 2025

ALEJANDRO SAINZ SAINZ
CLOUD COMPUTING

1 EXPLICACIÓN DE LA PRÁCTICA

Para esta práctica crearemos dos contenedores como en prácticas anteriores. Además de ello crearemos dos redes en docker. Luego, cada contenedor será asignado a una de las redes. Después realizaremos una serie de pruebas de conectivida.

2 COMENZANDO

Lo primero es arrancar la MV. Después de ahí, los primeros pasos serán crear las redes dentro de docker. Docker puede crear redes de la misma forma que azure, no sé si subredes, no he leído tanto. De todas formas, las redes creadas, aún alojadas en la misma máquina local son totalmente independientes e ignorantes la una de la otra. Pero bueno, vamos a lo que vamos.

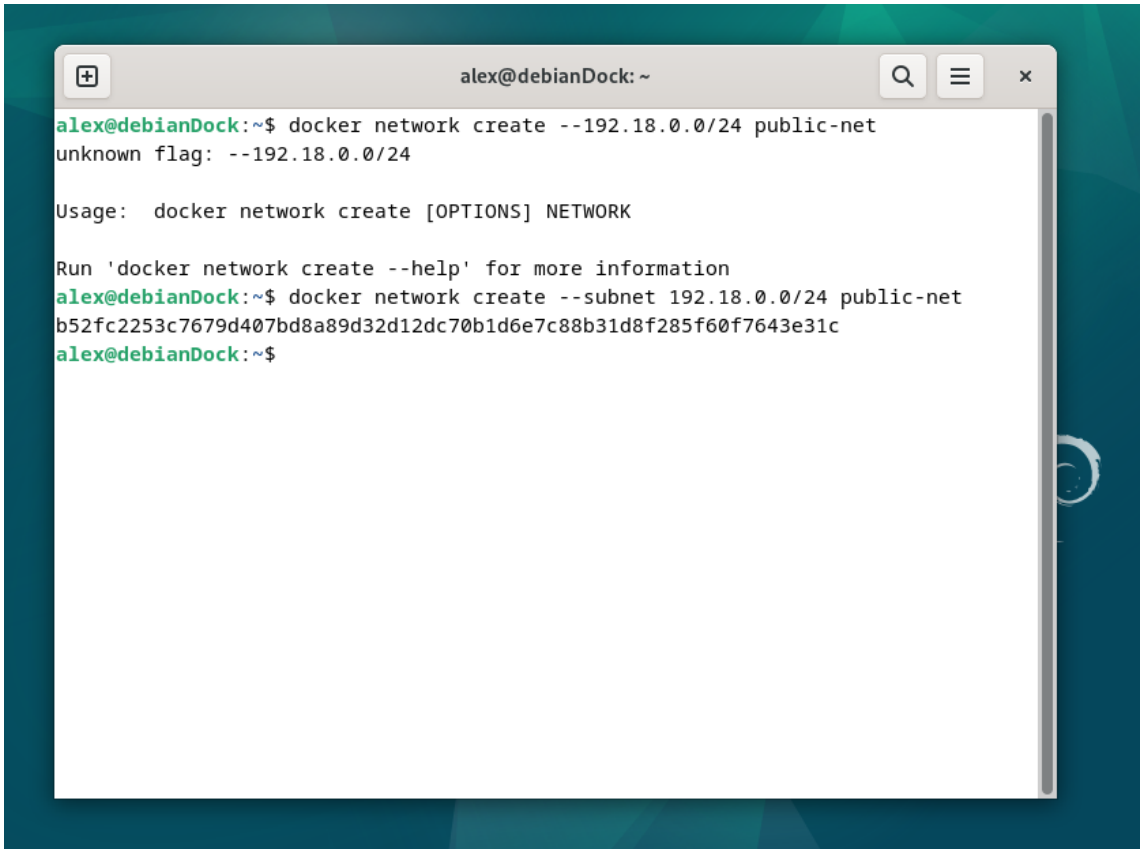

Ilustración 1 Creación de public-net

Creamos public-net con el comando que nos indica la práctica.

```
alex@debianDock:~$ docker network create --subnet 192.19.0.0/24 private-net
ade3fe57d265c7d9d38ba20f5defc9a303356d8f53e5fde4999c76513a498879
alex@debianDock:~$
```

Ilustración 2 Creación de private-net

Hacemos lo mismo para la red privada.

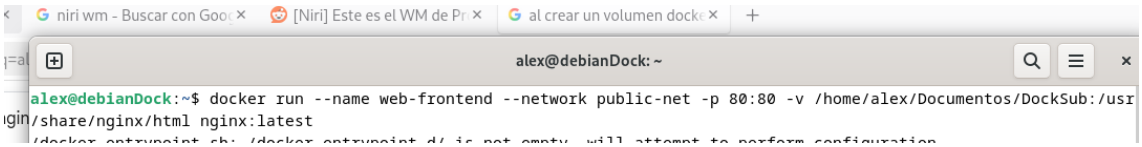

Ilustración 3 Creación del contenedor web-frontend

Usamos el mismo comando que hemos usado otras veces, pero añadiendo –network public-net antes de exponer los puertos. Con esto añadimos directamente este contenedor a esa red.

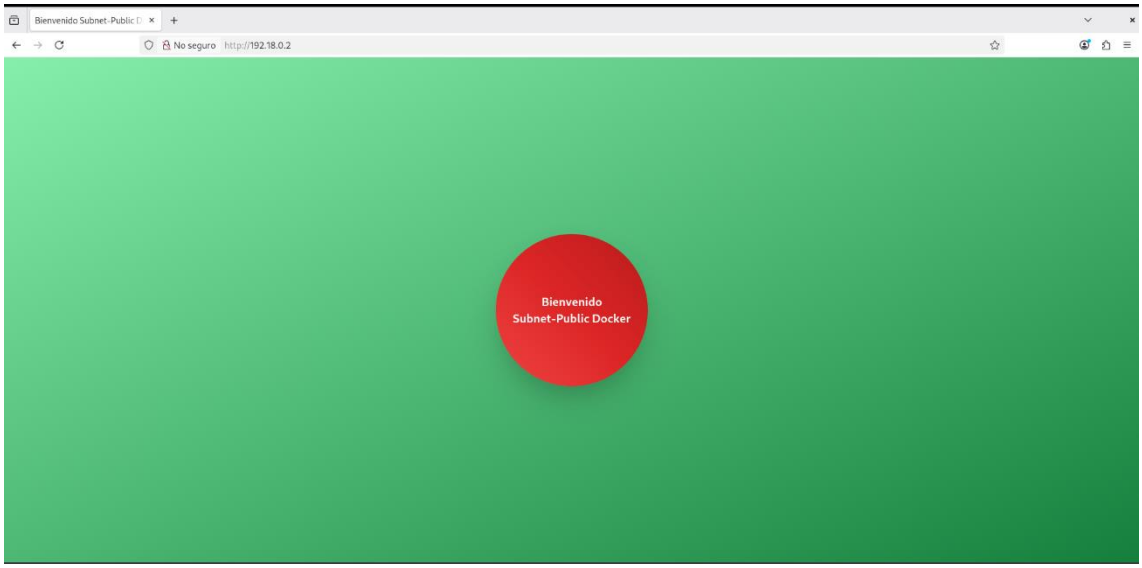

Ilustración 4 Comprobamos el contenedor

Comprobamos que el contenedor está levantado. Podría hacerlo con docker ps, pero si lo hago con el navegador accediendo a la web alojada queda mejor.

Ilustración 5 Comprobando las direcciones ip

A mí el comando que me daban en la práctica para comprobar la IP no me funcionaba, tampoco una variante con grep que encontré por ahí. Así que busque, en DockerDocs, y aparece este comando que vemos arriba: docker network inspect public-net. El resultado es más farragoso de ver, pero la verdad es que viene muy completo. Si bien es cierto, suele solo decirte cuál es su red y su Gateway, pero a veces es difícil encontrar la ip asignada, algunas veces está un poco escondida. Aunque en los dos casos fue la ip 2.

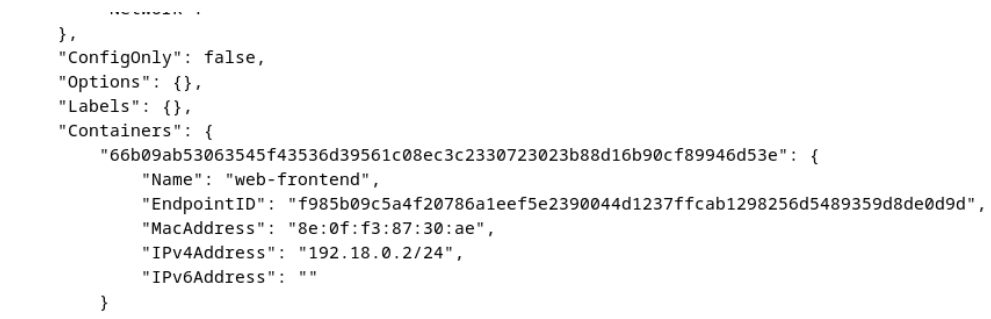

Ilustración 1 IP del contenedor

Luego, debemos comprobar las IPs de los contenedores. Como bien podemos ver aquí, nos ha asignado la ip 2.

Ilustración 2 Cosas de la terminal

Esta vez por alguna razón, al arrancar o crear contenedores, algunas veces el proceso me consume una pestaña entera de la terminal. Cuando mediante el navegador intento acceder al servidor web, por esa pestaña de la terminal me va dando avisos de errores y notificaciones. Así también puedo comprobar que el contenedor tiene actividad.

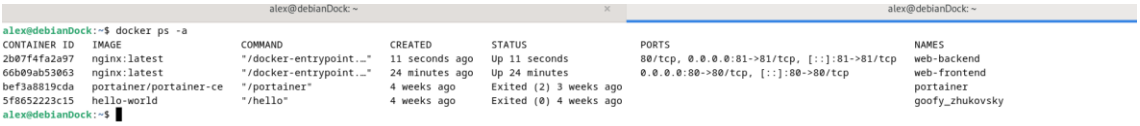

Ilustración 3 Repetimos proceso

Una vez realizadas todas las tareas de creación y comprobación del primer contenedor procedemos al segundo. Aquí se dio un caso extraño, creo yo. Como se ve en la imagen, para el segundo contenedor, tuve que exponer el puerto 81, me daba mensaje de error con el puerto 80. No sé si es por algo que hice mal, pero yo pensaba que al ser contenedores distintos e interfaces de red distintas no compartirían puerto, pero parece que sí. También se me hace raro que no me muestre la ip de los contenedores, solo la 0, aunque como podemos ver el contenedor Portainer tampoco la muestra, ni su puerto.


```
            "IPRange": "",
            "Gateway": "192.19.0.1"
        }
    ]
},
"Internal": false,
"Attachable": false,
"Ingress": false,
"ConfigFrom": {
    "Network": ""
},
"ConfigOnly": false,
"Options": {},
"Labels": {},
"Containers": {
    "2b07f4fa2a97d0b07607e9a7047b6228fb3cbd6caa79e8098db2acd2141e85f4": {
        "Name": "web-backend",
        "EndpointID": "69b82bf4481920e3711d179db9fd11eb5df0d47cee475bda55e3d3f41ef69cbe",
        "MacAddress": "12:d9:f2:bf:43:a4",
        "IPv4Address": "192.19.0.2/24",
        "IPv6Address": ""
    }
},
"Status": {
    "IPAM": {
        "Subnets": {
            "192.19.0.0/24": {
                "IPsInUse": 4,
                "DynamicIPsAvailable": 252
            }
        `
```


Ilustración 4 Inspeccionando web-backend

Aquí podemos ver la dirección del contenedor de backend. También le asigna la 2.

Ilustración 5 Otra cosa rara

Como el segundo contenedor también es nginx, y tiene un archivo html, intenté acceder por el puerto 81, y daba error, decía no encontrado. Supongo que quizá tenía que dar algún paso más, no estoy seguro. Sin embargo, con localhost, accedíamos a la página de bienvenida del servidor nginx. Esto se me ha atascado, y no termino de entender porque ha pasado así.

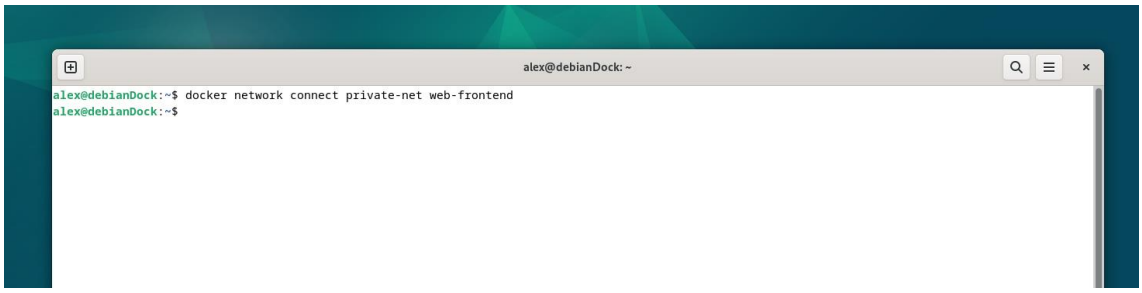

Ilustración 6 Vamos a conectar

Bueno, el caso es que ya tenemos dos contenedores, con dos redes. Haces ping de uno a otro y no se ven. Como ya dijimos, son independientes e ignorantes el uno del otro. Cada uno a lo suyo. Tienen salida al exterior, pero no se comunican entre ellos. Por lo tanto, busqué soluciones, y navegando acabé, y es lo normal, en DockerDocs. Hay varias formas de hacerlo. Yo he usado la que aparece en la imagen. Consiste en añadir una interfaz de red a uno de los contenedores que pertenezca a la red a la que no estaba unido. Es decir, uno de los contenedores tiene una ip por cada una de las redes creadas. Con eso se permite la comunicación.

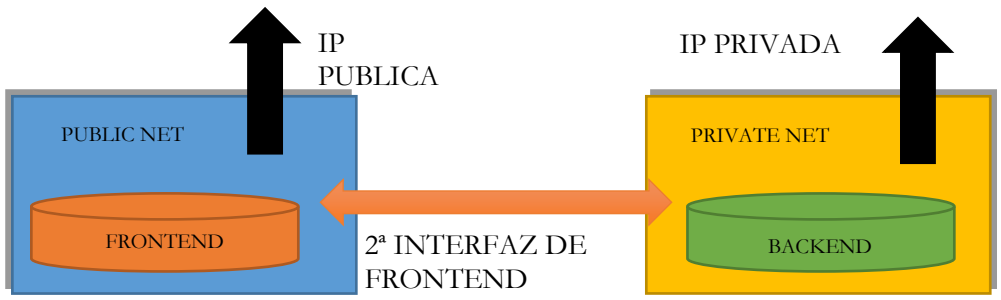

Ilustración 7 Diagrama de la situación

```
alex@debianDock:~$ docker inspect web-frontend
[
```

Ilustración 8 Alguna cosilla más

Evidentemente, no me iba el ping todavía. Cosas del directo. Así que de nuevo hay que comprobar que todo está en su sitio. Sobre todo, ver si frontend dispone ahora de dos interfaces de red.


```
        ]
    },
    "Networks": {
        "private-net": {
            "IPAMConfig": {
                "IPv4Address": "",
                "IPv6Address": ""
            },
            "Links": null,
            "Aliases": [],
            "DriverOpts": {},
            "GwPriority": 0,
            "NetworkID": "ade3fe57d265c7d9d38ba20f5defc9a303356d8f53e5fde4999c76513a498879",
            "EndpointID": "b6d6c3d0942f9d81a1c7f8f5dbbc95901d5ea229c9668f53e28340cfe4a911ae",
            "Gateway": "192.19.0.1",
            "IPAddress": "192.19.0.2",
            "MacAddress": "fe:b2:63:ed:83:f9",
            "IPPrefixLen": 24,
            "IPv6Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "DNSNames": [
                "web-frontend",
                "66b09ab53063"
            ]
        },
        "public-net": {
            "IPAMConfig": null,
            "Links": null,
            "Aliases": null,
            "DriverOpts": null,
            "GwPriority": 0,
            "NetworkID": "b52fc2253c7679d407bd8a89d32d12dc70b1d6e7c88b31d8f285f60f7643e31c",
            "EndpointID": "553044d619e8d5188345db6739e5a33531ea6f58e364c9a12068d1c198aa36ff",
            "Gateway": "192.18.0.1",
            "IPAddress": "192.18.0.2",
            "MacAddress": "2a:8b:da:d9:14:78",
            "IPPrefixLen": 24,
            "IPv6Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
```


Ilustración 9 Ahora sí que es farragoso

Como frontend dispone ahora de dos interfaces de red, el doble de líneas al desglosar sus componentes.

```
Error response from daemon: No such container: my_nginx
alex@debianDock:~$ docker container exec -it web-frontend /bin/bash
root@66b09ab53063:/# sudo apt install inetutils-ping
bash: sudo: command not found
root@66b09ab53063:/# apt-get install inetutils-ping
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package inetutils-ping
root@66b09ab53063:/#
```

Ilustración 10 Insalando Ping

Aquí el intento de instalar ping. Con el comando que vemos arriba, docker container exec –it web-frontend /bin/bash lo que estamos haciendo es acceder a la terminal del contenedor web-frontend, que es el que tiene las dos interfaces web. Y aquí mi erro, se me olvido hacer apt-get update o apt-get upgrade. Por eso me daba el error de paquete no encontrado, y no sabia que hacer porque sin esto no me reconoce el comando ping desde la terminal.

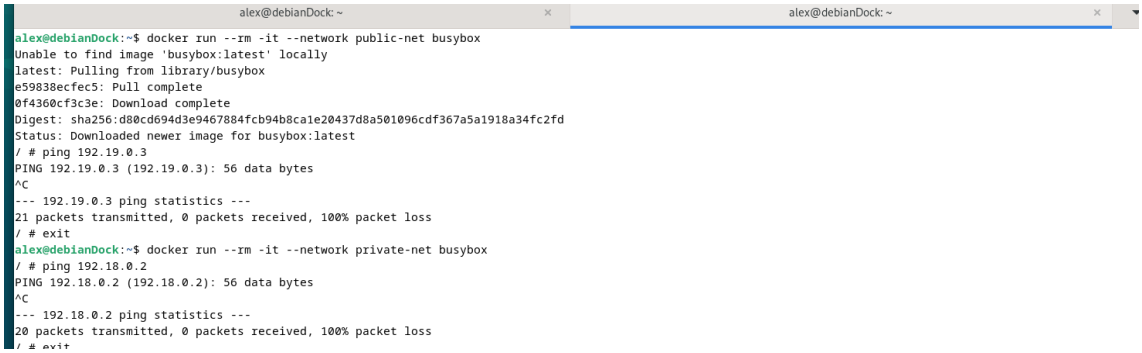

Ilustración 11 Un poco de ayuda

Pero bueno, gracias al señor David, aquí a mi derecha, lo que hicimos fue añadir busybox, que según el me explicó, es un paquete de herramientas para realizar este tipo de tareas, lo que nos permite, una vez ejecutado, ejecutar el comando ping.

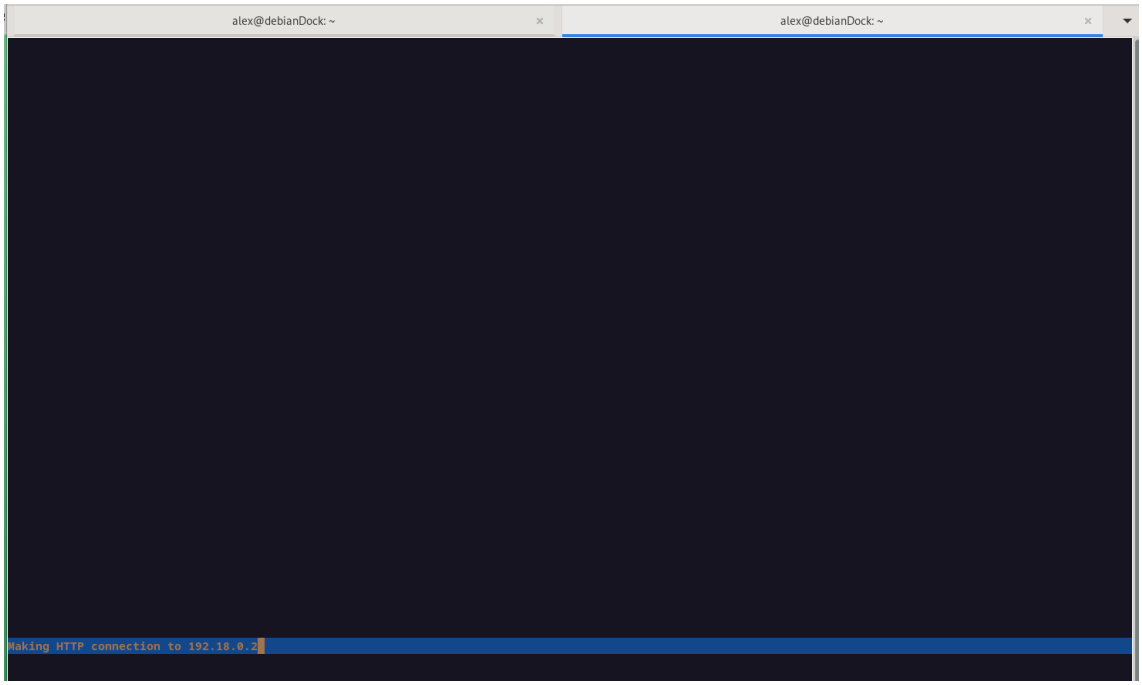

Ilustración 12 El toque personal

Cuando me di cuenta de que hubiese necesitado usar apt-get update, pude probar otra cosa, que es lo que vemos aquí. Se puede descargar el paquete Lynx. Es un paquete que nos permite acceder a una dirección web desde el terminal. Aquí la cosa era, que, aunque fuese en modo consola, intentar ver la página de front desde back.

Como siempre hago algo a medias, me di cuenta de que trataba de entrar a la interfaz web que no era. Así que después, hago un intento, pero tratando de comunicarme con la dirección de la Gateway de la red de front, public.

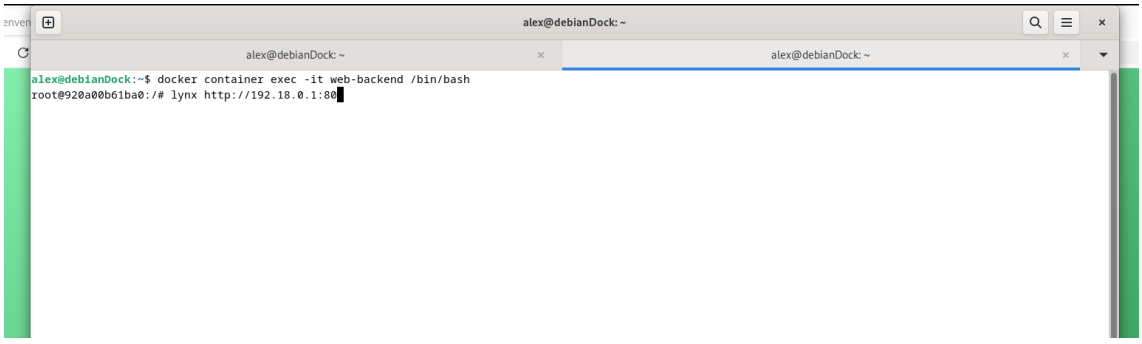

Ilustración 13 Ahora ya la cosa bien hecha.

Entro a la terminal de back y, desde ella, ejecuto Lynx hacia la ip de la Gateway de la red de front.

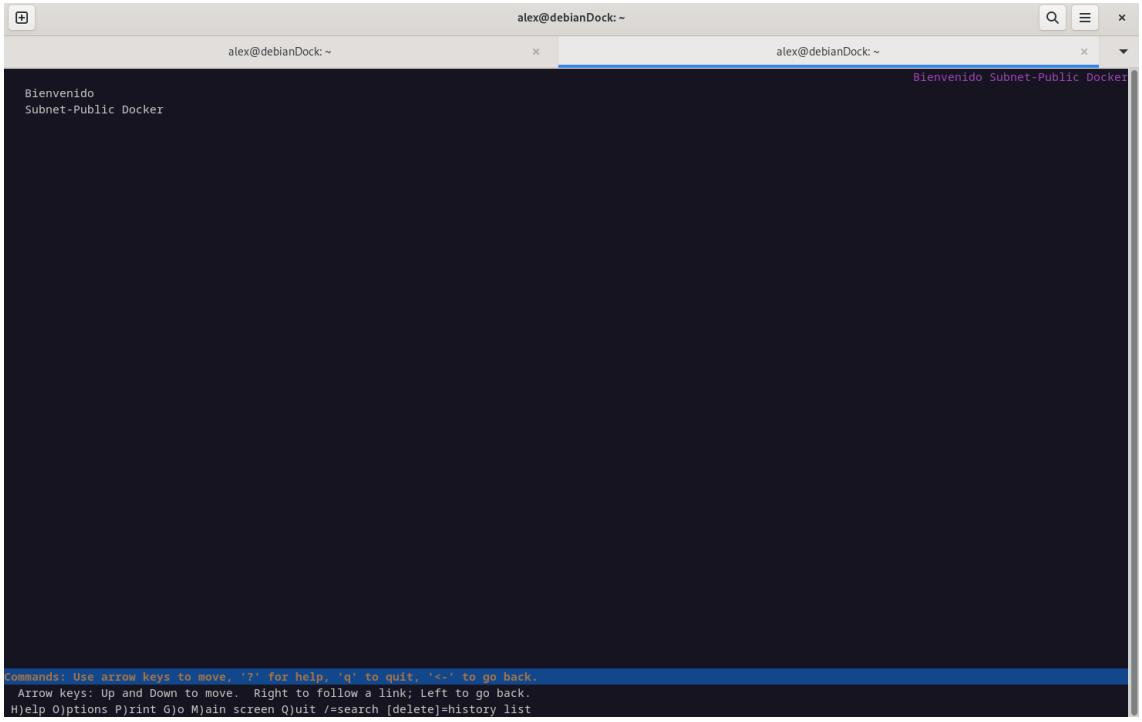

Ilustración 14 Parece lo mismo, pero no lo es

Y aquí está, de un modo muy rudimentario, pero podemos ver el mismo texto que en la web. Eso sí, de estilos nada, no se le puede pedir más a la terminal.

Con esto, damos por terminada la práctica.

3 DETALLES ADICIONALES

Como apunte breve, comentar que yo he elegido la opción más básica para conectar mis contenedores por red. He visto que hay otras formas en las que se puede crear un contenedor que funcione como router, pero, en ese caso, debo de hacer una configuración de red más completa, con port-forwarding y otras cosas de las que no entiendo mucho. Además, también he descubierto que a la hora de crear un contenedor también se le pueden añadir parámetros adicionales al comando para que se instalen ciertas herramientas en el momento de su creación. Por desgracia, lo descubrí a lo largo de la práctica, después de crear los contenedores.