CICLO FORMATIVO DE GRADO SUPERIOR - TÉCNICO EN ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN REDES

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS

Docker

Nombre y apellidos:
-Wuke Zhang

PRÁCTICA 4. Redes en Docker

Aunque no lo parezca, hasta ahora todos los contenedores que hemos utilizado han operado en una red virtual operada por docker, permitiendo la conexión de la máquina con el exterior y mapeo de puertos.

0. Eliminar todos los contenedores.

   Para eliminar todos los contenedores y así limpiar el entorno de prácticas, podemos utilizar una combinación de 2 comandos que ya hemos visto:

```
root@iso:/home/ubuntu/Desktop# docker rm -f $(docker ps -a -q)
```

```
To delete all Docker containers, I can run the Docker command that forces the removal of all containers. This command is not read-only because it alters the system state by removing containers.

✓ docker rm $(docker ps -aq) --force

All Docker containers have been successfully removed. If you have any other questions or need further assistance, feel free to ask!
```

   De la misma manera, si utilizamos docker ps --filter status=exited -q, podemos filtrar las maquinas a eliminar.

1. Ver la configuración de red de docker.

   Primero, levantamos un contenedor de Ubuntu en modo interactivo, como la imagen de Ubuntu contiene una distribución básica, instalaremos el comando ping para la prueba:

```
root@iso:/home/ubuntu/Desktop/web# docker run -it ubuntu
root@6a01c31c8409:/# apt-get update -y; apt-get install iputils-ping -y
```

```
bash
/mnt/c/Users/34640/Desktop (11.851s)
docker run -it --name ubuntu_test ubuntu bash

Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
5a7813e071bf: Download complete
Digest: sha256:72297848456d5d37d1262630108ab308d3e9ec7ed1c3286a32fe09856619a782
Status: Downloaded newer image for ubuntu:latest
root@0a954b5586b5:/# [ -z $WARP_BOOTSTRAPPED ] && printf "\\e]9278;f;{\"hook\": \"InitSubshell\", \"value\": { \"shell\": \"%s\", \"uname\": \"%s\" }}\\a" $([ $FISH_VERSION ] && echo "fish" || { echo $0 | command -p grep -q zsh && echo "zsh"; } || { echo $0 | command -p grep -q bash && echo "bash"; } || echo "unknown") $(uname)
root@0a954b5586b5:/#  [ -z $WARP_BOOTSTRAPPED ] && eval 'export WARP_HONOR_PS1=0;  command -p stty raw;unset PROMPT_COMMAND;HISTCONTROL=ignorespace;HISTIGNORE=" *";WARP_IS_SUBSHELL=1;WARP_SESSION_ID="$(command -p date +%s)$RANDOM";_hostname=$(command -pv hostname >/dev/null 2>&1 && command -p hostname 2>/dev/null || command -p uname -n);_user=$(command -pv whoami >/dev/null 2>&1 && command -p whoami 2>/dev/null || echo $USER);_msg=$(printf "{\"hook\": \"InitShell\", \"value\": {\"session_id\": $WARP_SESSION_ID, \"shell\": \"bash\", \"user\": \"$_user\", \"hostname\": \"$_hostname\", \"is_subshell\": true}}" | command -p od -An -v -tx1 | command -p tr -d " \n");if [[ "$OSTYPE" == "msys" ]]; then WARP_IS_GIT_BASH=true; else WARP_IS_GIT_BASH=false; fi;WARP_USING_WINDOWS_CON_PTY=true;if [ "$WARP_USING_WINDOWS_CON_PTY" = true ]; then printf '"'"'\e]9278;d;%s\x07'"'"' $_msg; else printf '"'"'\e\x50\x24\x64%s\x9c'"'"' $_msg; fi;unset _hostname _user _msg'
```

Prueba a hacer ping a Google.es, ¿Cuál es la dirección ip del contenedor (hostname -I)?

```
root@0a954b5586b5 / (3.117s)
ping -c 4 google.es

PING google.es (142.250.201.67) 56(84) bytes of data.
64 bytes from 142.250.201.67: icmp_seq=1 ttl=63 time=51.2 ms
64 bytes from 142.250.201.67: icmp_seq=2 ttl=63 time=49.7 ms
64 bytes from 142.250.201.67: icmp_seq=3 ttl=63 time=49.6 ms
64 bytes from 142.250.201.67: icmp_seq=4 ttl=63 time=49.5 ms

--- google.es ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 49.511/50.019/51.223/0.699 ms

root@0a954b5586b5:/ (0.021s)
hostname -I

172.17.0.2
```

2. Red por defecto, tipos de redes.

   Docker es capaz de gestionar múltiples redes virtuales de diferente tipo. Por defecto, siempre tiene activa tres redes específicas, compruébalo con el siguiente comando:

```
root@iso:/home/ubuntu/Desktop/web# docker network ls
```

```
bash
/mnt/c/Users/34640/Desktop (0.071s)
docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
e7fcc671bf3c   bridge    bridge    local
1bb19877288a   host      host      local
888252dfd1c2   none      null      local
```

   Si te fijas tenemos 3 redes distintas:

   - Bridge: por defecto alberga los contenedores que iniciamos, usando una red configurada con la ip 172.17.0.0/16. Los contenedores que creamos se conectan a esta red por defecto.
   - Host: los contenedores conectados a esta red ofrecen sus servicios directamente como el ordenador que los aloja, haciendo uso de su misma dirección ip y puertos.
   - None: red aislada del exterior que no configura ninguna ip, solamente loopback. Sirve para alojar contenedores que realicen tareas de segundo plano sobre volúmenes/directorios compartidos por otros ordenadores.

   Para ver la configuración de las redes, puedes ejecutar el siguiente comando:

```
docker network inspect $(docker network ls -q)
```

3. Gestión de redes en docker.

   Para crear redes en docker, disponemos del comando docker network create.

```
root@iso:/home/ubuntu/Desktop/web# docker network create red1
```

   Para ver las características de la red, emplea el comando docker network inspect.

```
root@iso:/home/ubuntu/Desktop/web# docker inspect red1
```

```
/mnt/c/Users/34640/Desktop (0.08s)
docker network inspect $(docker network ls -q)
[
    {
        "Name": "bridge",
        "Id": "e7fcc671bf3c33548700962c4495e46e92d6d7ee48ec967589d68ae9d7a378ac",
        "Created": "2025-02-21T23:18:35.137907479Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    },

/mnt/c/Users/34640/Desktop  Ctrl I  Agent Mode
```

Si te fijas esta vez a usado la dirección ip 172.18.0.0/16, es decir las va asignando incrementalmente. Para configurar manualmente una red, ejecuta la siguiente variante de docker network créate:

```
root@iso:/home/ubuntu/Desktop/web# docker network create
-d bridge --subnet 10.1.0.0/16 --gateway 10.1.0.1 red2
```

```
/mnt/c/Users/34640/Desktop (0.116s)
docker network create -d bridge --subnet 10.1.0.0/16 --gateway 10.1.0.1 red2
5b1c0d9eb303e2d981698eae91a50d1525ce66e1c1b526f3464fa95bdf879e19
```

Si te fija configuramos el tipo de red (-d), la ip y puerta de enlace.

Ahora bien, como su nombre indica, cada red que configuramos genera a su vez un puente de red como podemos comprobar al ejecutar ip a:

```
root@iso:/home/ubuntu/Desktop/web# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:97:c0:61 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 75555sec preferred_lft 75555sec
    inet6 fe80::170b:143c:563f:cb57/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:ec:45:db:3f brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:ecff:fe45:db3f/64 scope link
       valid_lft forever preferred_lft forever
12: br-df9876fe8578: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:3e:74:14:f6 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global br-df9876fe8578
       valid_lft forever preferred_lft forever
14: br-08f3c6cb2c27: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:01:ab:1b:63 brd ff:ff:ff:ff:ff:ff
    inet 10.1.0.1/16 brd 10.1.255.255 scope global br-08f3c6cb2c27
       valid_lft forever preferred_lft forever
```

```
/mnt/c/Users/34640/Desktop (0.056s)
docker network inspect red2

[
    {
        "Name": "red2",
        "Id": "5b1c0d9eb303e2d981698eae91a50d1525ce66e1c1b526f3464fa95bdf879e19",
        "Created": "2025-02-22T01:44:46.444807386Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "10.1.0.0/16",
                    "Gateway": "10.1.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

Para eliminar una red concreta, disponemos del comando:

```
root@iso:/home/ubuntu/Desktop/web# docker network remove red2
```

```
/mnt/c/Users/34640/Desktop (0.308s)
docker network rm red2
red2
```

Asimismo, para purgar las redes sin uso disponemos del comando:

```
root@iso:/home/ubuntu/Desktop/web# docker network prune
```

4. Uso de la red bridge por defecto.

   Aunque no es recomendable, podemos utilizar la red por defecto para enlazar diferentes contenedores y así hacer llamadas entre diferentes servicios. **En entornos profesionales, siempre tendremos que crear y configurar redes nuevas para usarlas.**

   Primero, crea un contenedor con base de datos, por ejemplo:

```
root@iso:/home/ubuntu/Desktop/web# docker run -d --name DB_server
-e MYSQL_ROOT_PASSWORD=ejemplo mysql
```

   Ahora, crearemos un servidor httpd vinculando la base de datos de la siguiente forma:

```
root@iso:/home/ubuntu/Desktop/web# docker run -d -
--name WEB_server --link DB_server:mysql httpd
```

```
/mnt/c/Users/34640/Desktop (10.358s)
docker run -d --name WEB_server --link DB_server:mysql httpd

Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
80350326cd93: Download complete
4f4fb700ef54: Download complete
a1a1b409f475: Download complete
830a84f99cc8: Download complete
35b1ecb71608: Download complete
c29f5b76f736: Download complete
Digest: sha256:3195404327ecd95b2fa0a5d4eac1f2206bb12996fb2561393f91254759e422b9
Status: Downloaded newer image for httpd:latest
323fda8db0d987ce0f3dc8f913c95818a13e2626e1d81fdddeab6bc0eb630122
```

   Si ejecutamos docker inspect sobre la red por defecto, los contenedores se encuentran registrados en la red correctamente.

```
root@iso:/home/ubuntu/Desktop/web# docker inspect bridge
```

```
},
"ConfigOnly": false,
"Containers": {
    "2b57302a4dbd7c7f16c7e86ba39d016e2a8a5c1a771f3b4236876fd2a868c7da": {
        "Name": "DB_server",
        "EndpointID": "58cebf3ad443bb7a31b81147b378c2e4bb03ffe65463c80c21dc450c581f67c0",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
    },
    "323fda8db0d987ce0f3dc8f913c95818a13e2626e1d81fdddeab6bc0eb630122": {
        "Name": "WEB_server",
        "EndpointID": "1b3a459a795c4c33f272c3078cd1ebf162bce76dd311370dc52b691e56e282d5",
        "MacAddress": "02:42:ac:11:00:03",
        "IPv4Address": "172.17.0.3/16",
        "IPv6Address": ""
    }
},
"Options": {
```

Respecto a los propios contenedores, los efectos se producen principalmente en el contenedor que recibe la vinculación, el cual tiene acceso a la siguiente información:

- Direccion del contenedor vinculado:

```
root@iso:/home/ubuntu/Desktop# docker exec WEB_server cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.2      mysql 72e6d1c570b2 DB_server
172.17.0.3      3a58613eebe1
```

```
/mnt/c/Users/34640/Desktop (0.137s)
docker exec WEB_server cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.2      mysql 2b57302a4dbd DB_server
172.17.0.3      323fda8db0d9
```

- Variables de entorno del contenedor vinculado (IMG_ENV_variabledeentorno)

```
root@iso:/home/ubuntu/Desktop# docker exec WEB_server env
PATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=3a58613eebe1
MYSQL_PORT=tcp://172.17.0.2:3306
MYSQL_PORT_3306_TCP=tcp://172.17.0.2:3306
MYSQL_PORT_3306_TCP_ADDR=172.17.0.2
MYSQL_PORT_3306_TCP_PORT=3306
MYSQL_PORT_3306_TCP_PROTO=tcp
MYSQL_PORT_33060_TCP=tcp://172.17.0.2:33060
MYSQL_PORT_33060_TCP_ADDR=172.17.0.2
MYSQL_PORT_33060_TCP_PORT=33060
MYSQL_PORT_33060_TCP_PROTO=tcp
MYSQL_NAME=/WEB_server/mysql
MYSQL_ENV_MYSQL_ROOT_PASSWORD=ejemplo
MYSQL_ENV_GOSU_VERSION=1.16
MYSQL_ENV_MYSQL_MAJOR=innovation
MYSQL_ENV_MYSQL_VERSION=8.2.0-1.el8
MYSQL_ENV_MYSQL_SHELL_VERSION=8.2.1-1.el8
HTTPD_PREFIX=/usr/local/apache2
HTTPD_VERSION=2.4.58
HTTPD_SHA256=fa16d72a078210a54c47dd5bef2f8b9b8a01d94909a51453956b3ec6442ea4c5
HTTPD_PATCHES=
HOME=/root
```

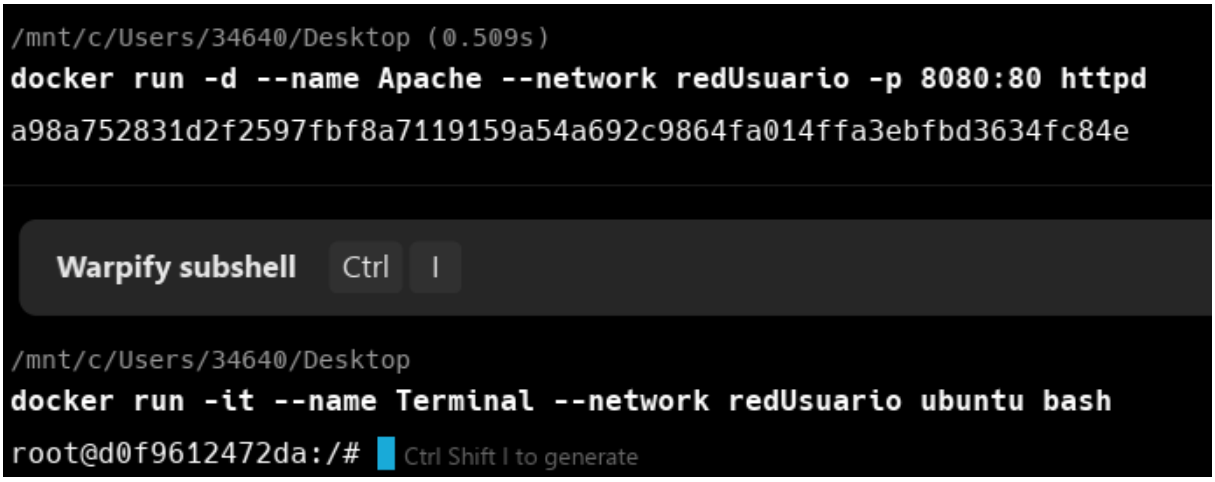
```
/mnt/c/Users/34640/Desktop (0.093s)
docker exec WEB_server env
PATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=323fda8db0d9
MYSQL_PORT=tcp://172.17.0.2:3306
MYSQL_PORT_3306_TCP=tcp://172.17.0.2:3306
MYSQL_PORT_3306_TCP_ADDR=172.17.0.2
MYSQL_PORT_3306_TCP_PORT=3306
MYSQL_PORT_3306_TCP_PROTO=tcp
MYSQL_PORT_33060_TCP=tcp://172.17.0.2:33060
MYSQL_PORT_33060_TCP_ADDR=172.17.0.2
MYSQL_PORT_33060_TCP_PORT=33060
MYSQL_PORT_33060_TCP_PROTO=tcp
MYSQL_NAME=/WEB_server/mysql
MYSQL_ENV_MYSQL_ROOT_PASSWORD=ejemplo
MYSQL_ENV_GOSU_VERSION=1.17
MYSQL_ENV_MYSQL_MAJOR=innovation
MYSQL_ENV_MYSQL_VERSION=9.2.0-1.el9
MYSQL_ENV_MYSQL_SHELL_VERSION=9.2.0-1.el9
HTTPD_PREFIX=/usr/local/apache2
HTTPD_VERSION=2.4.63
HTTPD_SHA256=88fc236ab99b2864b248de7d49a008ec2afd7551e64dce8b95f58f32f94c46ab
HTTPD_PATCHES=
HOME=/root
```

5. Uso de redes creadas por el usuario.

   Las redes creadas por el administrador del servidor son más adecuadas para las necesidades de los diferentes servicios. Esto se debe a que permiten conectar y desconectar la red del contenedor en caliente, mientras que **la red bridge por defecto no puede si el contenedor esta marcha.**

   Probemos a conectar dos contenedores a nuestra red, ejecutando los siguientes comandor:

   - docker network create redUsuario
   - docker run -d --name Apache --network redUsuario -p 8080:80 httpd
   - docker run -it --name Terminal --network redUsuario ubuntu bash
   -



   Dentro de la terminal bash donde nos encontramos, ejecutaremos el siguiente comando:

   - apt update; apt install dnsutils -y

   Ahora podemos realizar una búsqueda de dns, donde si apuntamos a Apache, obtenemos el siguiente resultado:

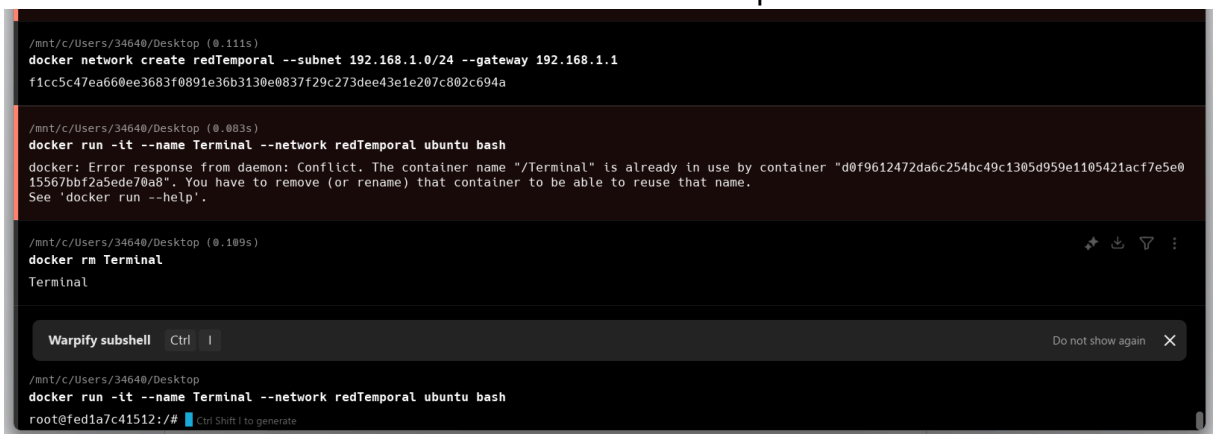
```
root@73cd9af0de8b:/# dig Apache

; <<>> DiG 9.18.18-0ubuntu0.22.04.1-Ubuntu <<>> Apache
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13058
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;Apache.                                IN      A

;; ANSWER SECTION:
Apache.                 600     IN      A       172.20.0.2

;; Query time: 0 msec
;; SERVER: 127.0.0.11#53(127.0.0.11) (UDP)
;; WHEN: Wed Jan 17 20:11:48 UTC 2024
;; MSG SIZE  rcvd: 46
```

```
Processing triggers for libc-bin (2.39-0ubuntu8.3) ...
root@d0f9612472da:/# dig apache

; <<>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <<>> apache
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53118
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;apache.                                IN      A

;; ANSWER SECTION:
apache.                 600     IN      A       172.18.0.2

;; Query time: 0 msec
;; SERVER: 127.0.0.11#53(127.0.0.11) (UDP)
;; WHEN: Sat Feb 22 01:56:26 UTC 2025
;; MSG SIZE  rcvd: 46

root@d0f9612472da:/#
```

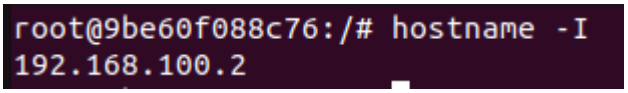
Ahora, crearemos una red nueva con un contededor ubuntu, mediante el siguiente comando (recuerda salir de la terminal):

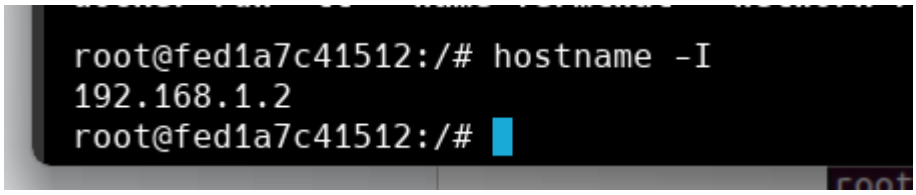
- docker network create redTemporal --subnet 192.168.100.0/24 --gateway 192.168.100.1

-docker run -it --name TerminaTMP --network redTemporal Ubuntu bash

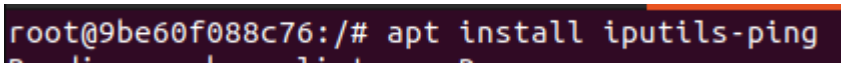


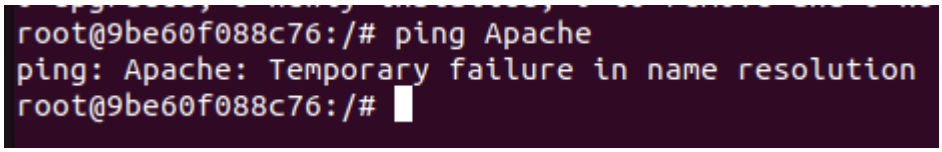
Si ejecutamos hostname -I, comprobamos como nos encontramos en la red que habíamos configurado:

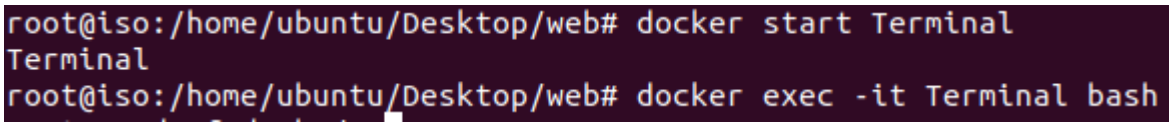


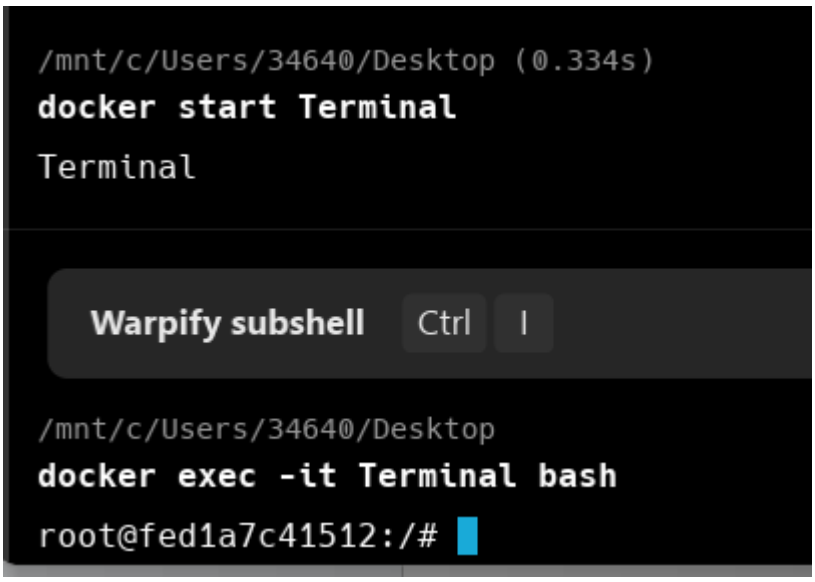


Además, tampoco tenemos conexión con los otros contenedores:









Probemos a conectar la Terminal a la red Temporal:

```
root@iso:/home/ubuntu/Desktop# docker network connect redTemporal Terminal
root@iso:/home/ubuntu/Desktop# docker network inspect redTemporal
```

```
/mnt/c/Users/34640/Desktop (0.189s)
docker network connect redTemporal Terminal

/mnt/c/Users/34640/Desktop (0.053s)
docker network inspect redTemporal
[
    {
        "Name": "redTemporal",
        "Id": "f1cc5c47ea660ee3683f0891e36b3130e0837f29c273dee43e1e207c802c694a",
        "Created": "2025-02-22T01:57:44.909113886Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "192.168.1.0/24",
                    "Gateway": "192.168.1.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },

/mnt/c/Users/34640/Desktop Ctrl I Agent Mode
```

Desde TerminalTMB, comprobamos que podemos hacer ping a Terminal, a través de redTemporal:

```
root@9be60f088c76:/# ping Terminal
PING Terminal (192.168.100.3) 56(84) bytes of data.
64 bytes from Terminal.redTemporal (192.168.100.3): icmp_seq=1 ttl=64 time=0.170 ms
```

Y desde Terminal a TerminalTMB:

```
root@73cd9af0de8b:/# ping TerminalTMB
PING TerminalTMB (192.168.100.2) 56(84) bytes of data.
64 bytes from TerminalTMB.redTemporal (192.168.100.2): icmp_seq=1 ttl=64 time=0.113 ms
```

```
/mnt/c/Users/34640/Desktop (7.678s)
ping 192.168.1.1

PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=63 time=0.925 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=63 time=1.80 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=63 time=1.65 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=63 time=0.912 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=63 time=1.14 ms
64 bytes from 192.168.1.1: icmp_seq=6 ttl=63 time=0.771 ms
64 bytes from 192.168.1.1: icmp_seq=7 ttl=63 time=0.833 ms

64 bytes from 192.168.1.1: icmp_seq=8 ttl=63 time=0.842 ms
^C
--- 192.168.1.1 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7185ms
rtt min/avg/max/mdev = 0.771/1.109/1.803/0.372 ms
```

6. Parámetros adicionales y creación simultanea de contenedores y redes.

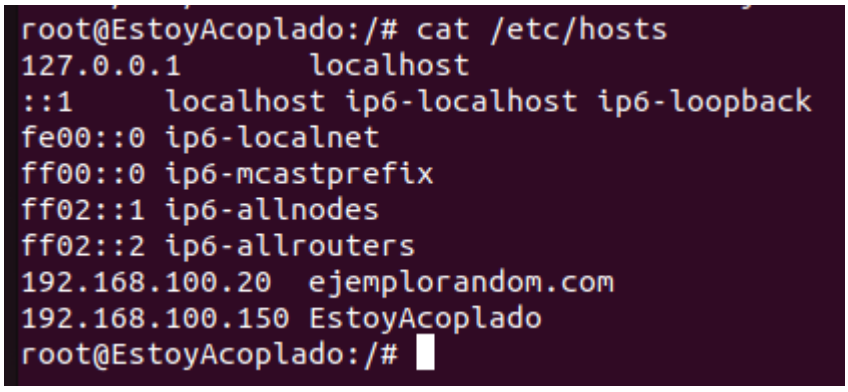
Adicionalmente a la ip y puerta de enlace, existen más opciones al trabajar con redes en docker. Al crear un contenedor, podemos utilizar las siguientes flafs para configurar los siguientes parámetros:

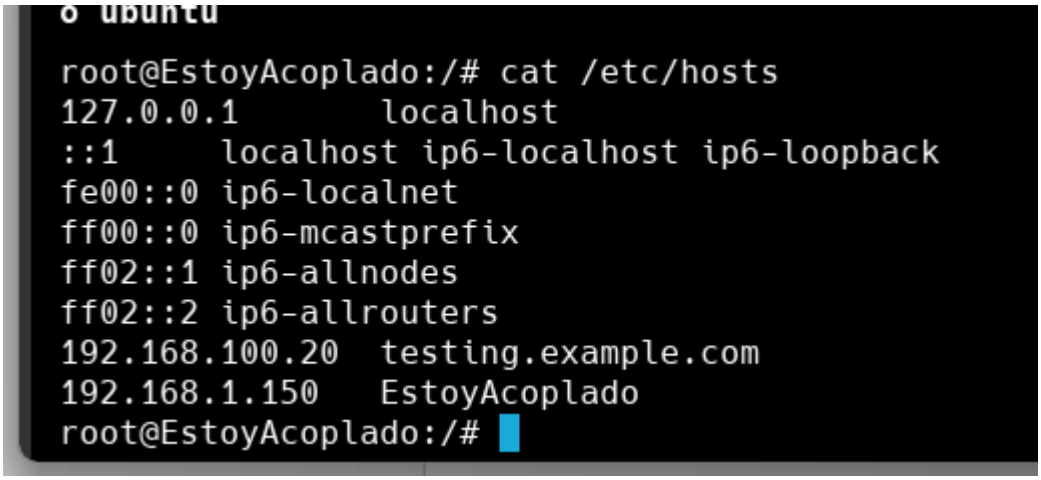
--dns: para establecer unos servidores DNS predeterminados.
--ip: Para establecer una ip fija en el contenedor.
--ip6: para establecer la dirección de red ipv6
--hostname o -h: para establecer el nombre de host del contenedor. Si no lo establezco será el ID del mismo.
--add-host: añade entradas de nuevos hosts en el fichero /etc/hosts

Un ejemplo de contenedor con esta configuración sería el siguiente, ejecútalo:

docker run -it --name Acoplado --network redTempolar --ip 192.168.100.150 --add-host=testing.example.com:192.168.100.20 --dns 8.8.8.8 --hostname EstoyAcoplado Ubuntu

Si vemos el archivo /etc/hosts, vemos como tanto nuestra maquina como el sitio añadido figuran correctamente.





Por último, si volvemos a ver el archivo /etc/resolv.conf, vemos como no se ha

actualizado el DNS...

```
root@EstoyAcoplado:/# cat /etc/resolv.conf
nameserver 127.0.0.11
options edns0 trust-ad ndots:0
```

```
root@EstoyAcoplado:/# cat /etc/resolv.conf
# Generated by Docker Engine.
# This file can be edited; Docker Engine will not make further changes once it
# has been modified.

nameserver 127.0.0.11
options ndots:0

# Based on host file: '/etc/resolv.conf' (internal resolver)
# ExtServers: [8.8.8.8]
# Overrides: [nameservers]
# Option ndots from: internal
root@EstoyAcoplado:/#
```

Esto se debe a que el DNS no se configura a nivel interno del contendor, sino que lo gestiona docker. Con docker inspect Acoplado vemos que la informacion esta correctamente registrada

```
root@iso:/home/ubuntu/Desktop# docker inspect Acoplado | tail -n 160
            "VolumeDriver": "",
            "VolumesFrom": null,
            "ConsoleSize": [
                32,
                108
            ],
            "CapAdd": null,
            "CapDrop": null,
            "CgroupnsMode": "private",
            "Dns": [
                "8.8.8.8"
            ],
            "DnsOptions": [],
            "DnsSearch": [],
            "ExtraHosts": [
                "ejemplorandom.com:192.168.100.20"
            ],
            "GroupAdd": null,
```

```
/mnt/c/Users/34640/Desktop (0.064s)
docker inspect Acoplado | tail -n 160
            "AutoRemove": false,
            "VolumeDriver": "",
            "VolumesFrom": null,
            "ConsoleSize": [
                47,
                155
            ],
            "CapAdd": null,
            "CapDrop": null,
            "CgroupnsMode": "host",
            "Dns": [
                "8.8.8.8"
            ],
            "DnsOptions": [],
            "DnsSearch": [],
            "ExtraHosts": [
                "testing.example.com:192.168.100.20"
            ],
            "GroupAdd": null,
```