

# Trabajo Práctico

## Configuración inicial de la red del laboratorio

Fecha de entrega: 8/04/2021

Franco, Juan Martín 149.615

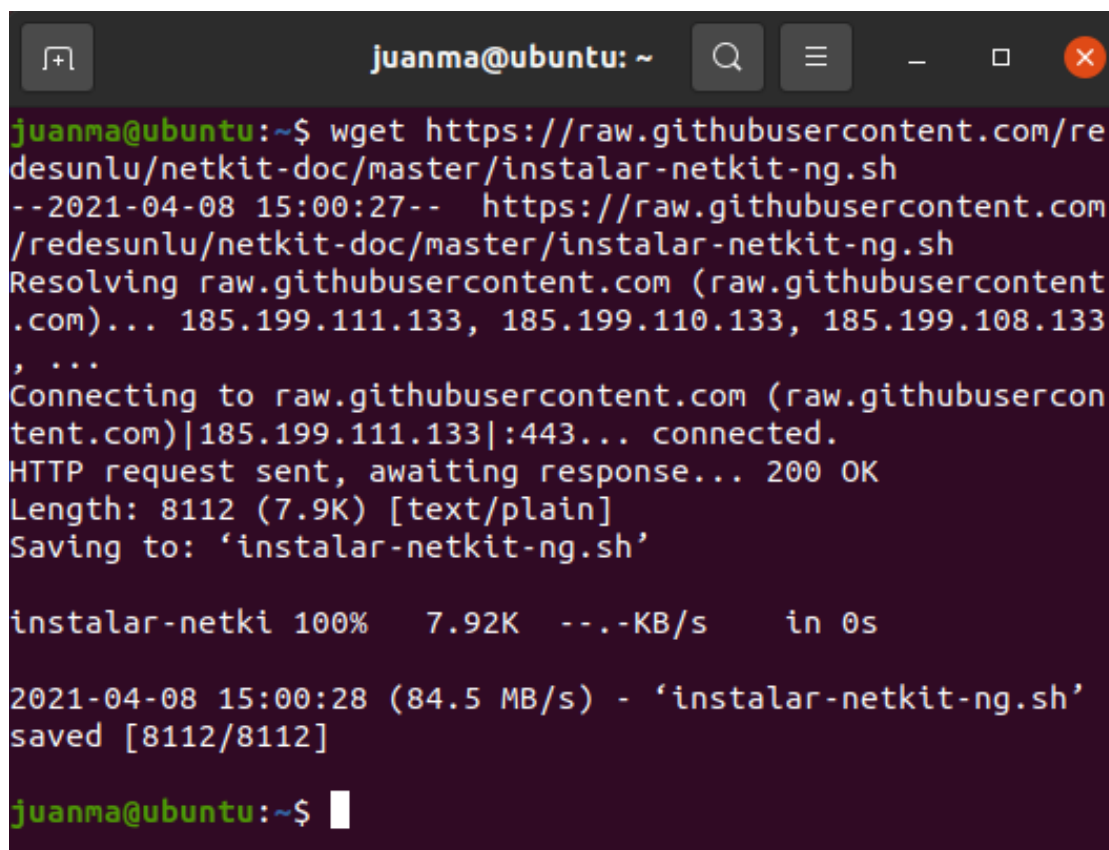
juanmartin\_franco@hotmail.com

### Primer parte: Instalación del entorno Netkit

Para dar comienzo con el trabajo práctico, instalaremos el laboratorio Netkit, el cual permite emular 2 máquinas virtuales.

Para comenzar con la descarga de dicho laboratorio, se procede a iniciar la máquina virtual (en mi caso, con Ubuntu instalado) y en la terminal escribo el siguiente comando:

```
wget https://raw.githubusercontent.com/redesunlu/netkit-doc/master/instalar-netkit-ng.sh
```



```
juanma@ubuntu: ~  
juanma@ubuntu:~$ wget https://raw.githubusercontent.com/redesunlu/netkit-doc/master/instalar-netkit-ng.sh  
--2021-04-08 15:00:27-- https://raw.githubusercontent.com/redesunlu/netkit-doc/master/instalar-netkit-ng.sh  
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.110.133, 185.199.108.133, ...  
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 8112 (7.9K) [text/plain]  
Saving to: 'instalar-netkit-ng.sh'  
  
instalar-netkit-ng.sh 100% 7.92K ---KB/s in 0s  
  
2021-04-08 15:00:28 (84.5 MB/s) - 'instalar-netkit-ng.sh' saved [8112/8112]  
  
juanma@ubuntu:~$
```

Una vez hecho esto, debemos ejecutar el script descargado recientemente con el comando:

```
bash instalar-netkit-ng.sh
```

Obteniendo el siguiente resultado:

```

juanma@ubuntu:~$ bash instalar-netkit-ng.sh
=====
=====
Esta secuencia de comandos instalará Netkit-NG en este
equipo.
La instalación requiere al menos 2 GB de espacio disponible.
Todos los archivos se almacenarán en el directorio /home/juanma/netkit

Verifique la documentación disponible en
https://github.com/redesunlu/netkit-doc
=====
=====

» Descargando netkit-ng desde el repositorio alternativo .
..
netkit-ng-core 100% 588.52K 1.76MB/s in 0.3s
netkit-ng-file 100% 517.94M 2.85MB/s in 3m 15s
netkit-ng-kern 100% 7.03M 2.77MB/s in 2.5s

» Verificando la integridad de los paquetes ... OK

```

Una vez se termina de descargar y descomprimir todos los archivos, se mostrará una pantalla como la siguiente, la cual indica que la instalación finalizó correctamente.

```

juanma@ubuntu: ~/netkit/netkit...
konsole      : not found
gnome-terminal : found
passed.
> Checking filesystem type... passed.
> Checking whether executables can run... passed.
> Checking for availability of dumper tools:
    wireshark    : ok
    tcpdump      : ok
    uml_dump     : ok
passed.
[ READY ] Congratulations! Your Netkit setup is now complete!
        Enjoy Netkit!

» Instalación finalizada.
Pruebe iniciar un laboratorio con

    cd
    cd /home/juanma/netkit/netkit-lab_webserver
    lstart

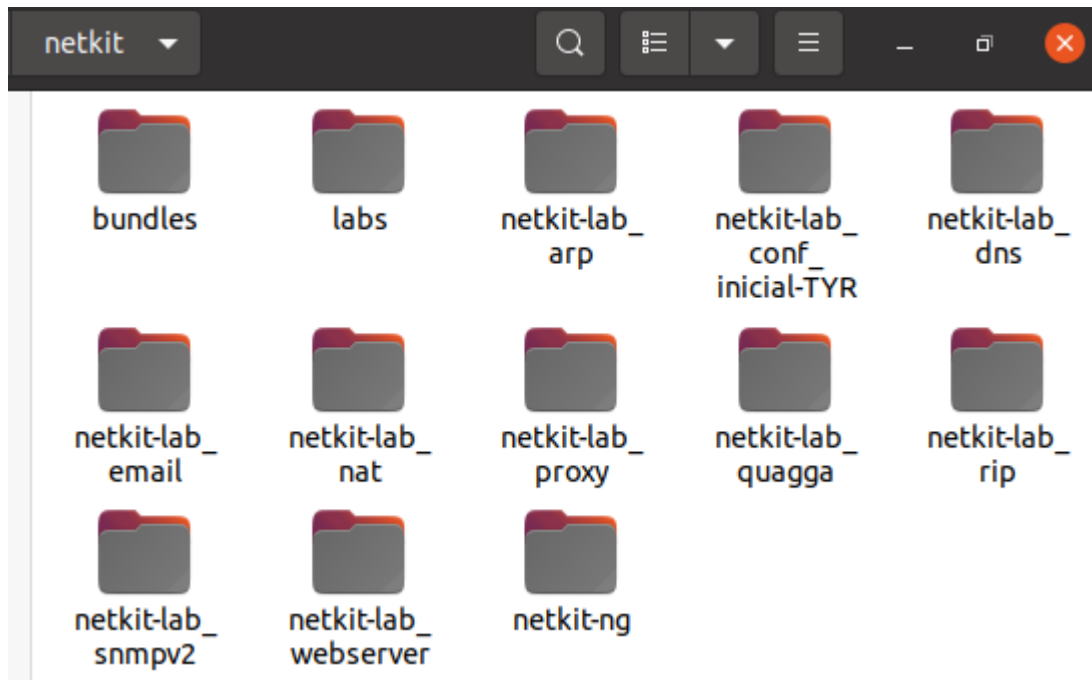
juanma@ubuntu:~/netkit/netkit-ng$

```

Una vez instalado el laboratorio, procedo con descargar el laboratorio inicial desde el siguiente link:

[https://github.com/redesunlu/netkit-labs/raw/master/tarballs/netkit-lab\\_conf\\_inicial-TYR.tar.gz](https://github.com/redesunlu/netkit-labs/raw/master/tarballs/netkit-lab_conf_inicial-TYR.tar.gz)

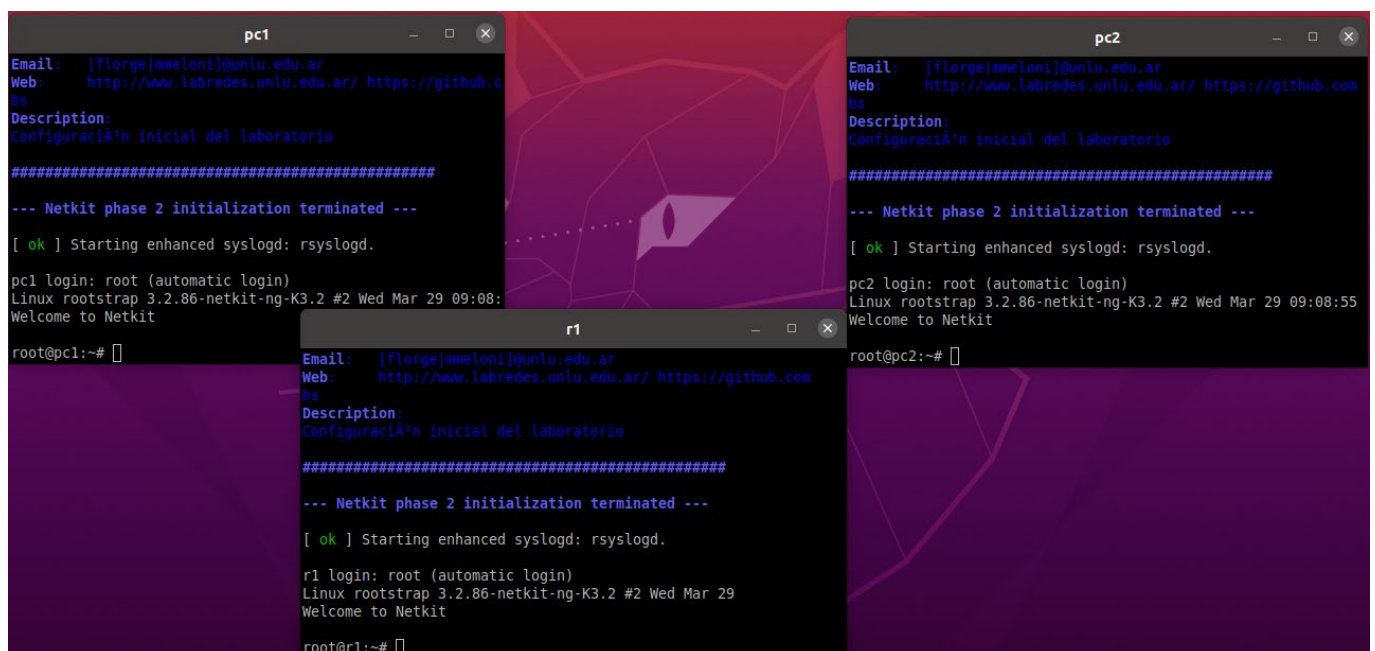
En mi caso, opté por guardarlo en la misma carpeta de instalación de Netkit.



Hecho esto, procedo a iniciar la ejecución del laboratorio posicionándome en la carpeta de este y utilizando el comando:

```
lstart
```

Una vez que ejecuto dicho comando, se inicia el laboratorio, obteniendo lo siguiente:



En dicha imagen se pueden apreciar las 2 máquinas virtuales (pc1 y pc2) y una tercera llamada r1.

En caso de querer finalizar con la ejecución del laboratorio, debo ejecutar el siguiente comando sobre la misma consola donde inicié el laboratorio (es decir, no lo debo ejecutar sobre ninguna máquina virtual del laboratorio):

```
lhalt
```

```
juanma@ubuntu:~/netkit/netkit-lab_conf_inicial-TYR/netkit-lab_conf_inicial$ lhalt

===== Halting lab =====
Lab directory: /home/juanma/netkit/netkit-lab_conf_inicial-TYR/netkit-lab_conf_inicial
Version:      1
Author:       F. Lorge, M. Meloni
Email:        [florge|mmeloni]@unlu.edu.ar
Web:          http://www.labredes.unlu.edu.ar/ https://github.com/redesunlu/netkit-labs
Description:  Configuración inicial del laboratorio
=====
Halting virtual machine "pc1" (PID 2340) owned by juanma [.....]
Halting virtual machine "pc2" (PID 3647) owned by juanma [.....]
Halting virtual machine "r1" (PID 5033) owned by juanma [.....]
Removing readyfor.test...

Lab has been halted.
=====
```

## Segunda parte: Configuración de hosts en una red - prueba de conectividad - análisis de captura

**1. Verificar la/s interfaces de red (comúnmente llamada placa de red o NIC) que el sistema operativo haya detectado en pc1 y pc2. Para ello debe primero activar las interfaces disponibles, y luego listar su información en pantalla.**

Para verificar las interfaces de red que el sistema operativo haya detectado debo utilizar el siguiente comando:

```
ip address show (o ip a s)
```

Obteniendo los siguientes resultados para cada pc:

```
pc1
bs
Description:
Configuración inicial del laboratorio

#####

--- Netkit phase 2 initialization terminated ---

[ ok ] Starting enhanced syslogd: rsyslogd.

pc1 login: root (automatic login)
Last login: Thu Apr  8 23:14:45 ART 2021 on tty0
Linux pc1 3.2.86-netkit-ng-K3.2 #2 Wed Mar 29 09:08:55 ART 2017 i686
Welcome to Netkit

root@pc1:~# ip a s
1: lo: <LOOPBACK,UP,LOWER UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether 12:4d:1b:48:43:91 brd ff:ff:ff:ff:ff:ff
root@pc1:~#
```

```
pc2
Linux pc2 3.2.86-netkit-ng-K3.2 #2 Wed Mar 29 09:08:55 ART 2017 i686
Welcome to Netkit

root@pc2:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether d2:95:d2:aa:7f:d2 brd ff:ff:ff:ff:ff:ff
```

Luego, para activar dichas interfaces, debo utilizar el comando:

```
ip link set eth0 up
```

Siendo eth0 el nombre de la interfaz que deseo activar.

Los resultados son los siguientes:

pc1:

```
root@pc1:~# ip link set eth0 up
root@pc1:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
    link/ether 12:4d:1b:48:43:91 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::104d:1bff:fe48:4391/64 scope link
        valid_lft forever preferred_lft forever
root@pc1:~#
```

pc2:

```
root@pc2:~# ip link set eth0 up
root@pc2:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
    link/ether d2:95:d2:aa:7f:d2 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::d095:d2ff:feaa:7fd2/64 scope link
        valid_lft forever preferred_lft forever
root@pc2:~#
```

Los nombres de las interfaces son:

lo (loopback)

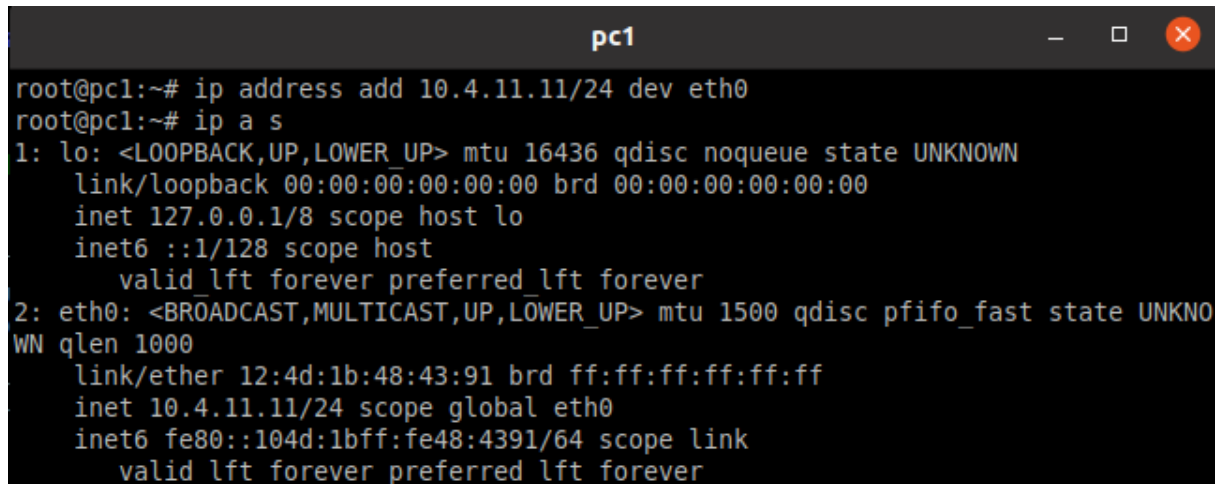
eth0

El nombre de la interfaz que se encuentra conectada a la red es eth0.

**2) Configuración de interfaces de red para utilizar el protocolo TCP/IP. El paso siguiente es asignar las direcciones IP 10.4.11.11 y 10.4.11.12 a pc1 y pc2 respectivamente (la máscara de red es /24 o 255.255.255.0).**

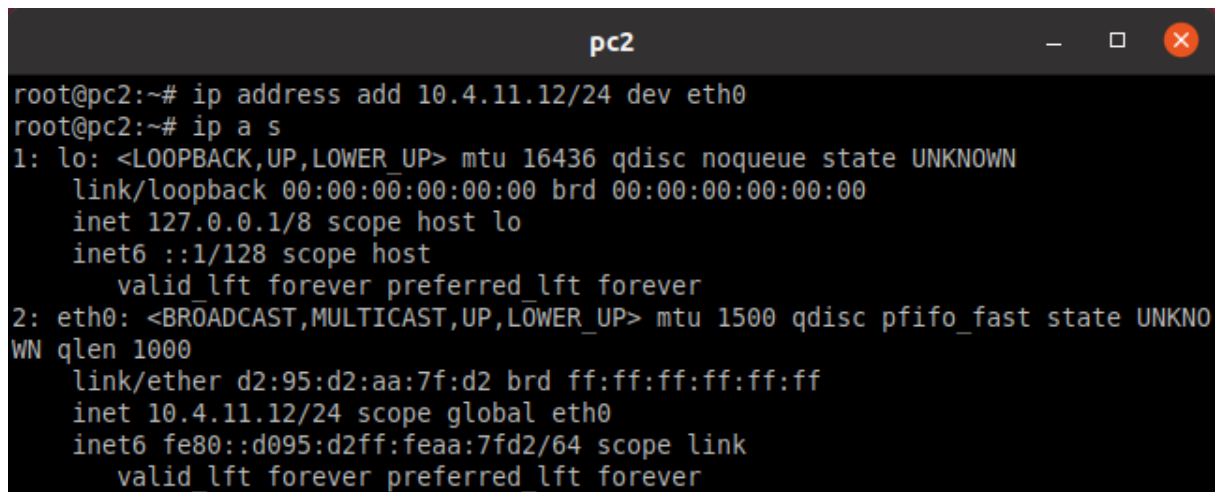
Para realizar dicha asignación, debo utilizar los siguientes comandos:

```
ip address add 10.4.11.11/24 dev eth0 (para pc1)
```



```
pc1
root@pc1:~# ip address add 10.4.11.11/24 dev eth0
root@pc1:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOW
    link/ether 12:4d:1b:48:43:91 brd ff:ff:ff:ff:ff:ff
    inet 10.4.11.11/24 scope global eth0
    inet6 fe80::104d:1b:48:43:91/64 scope link
        valid_lft forever preferred_lft forever
```

```
ip address add 10.4.11.12/24 dev eth0 (para pc2)
```



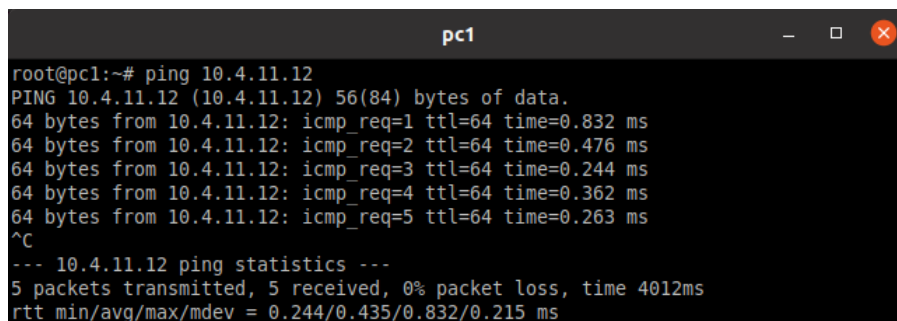
```
pc2
root@pc2:~# ip address add 10.4.11.12/24 dev eth0
root@pc2:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOW
    link/ether d2:95:d2:aa:7f:d2 brd ff:ff:ff:ff:ff:ff
    inet 10.4.11.12/24 scope global eth0
    inet6 fe80::d095:d2:aa:7f:d2/64 scope link
        valid_lft forever preferred_lft forever
```

**3) Verificar que es posible contactar ambos equipos de la red.**

Para verificar que es posible contactar ambos equipos de la red, basta con hacer un ping desde una pc hacia la otra y si recibo una respuesta quiere decir que fue posible contactarse.

Para hacer esto, utilizo el comando:

```
ping 10.4.11.12 (desde la pc1)
```



```
pc1
root@pc1:~# ping 10.4.11.12
PING 10.4.11.12 (10.4.11.12) 56(84) bytes of data.
64 bytes from 10.4.11.12: icmp_req=1 ttl=64 time=0.832 ms
64 bytes from 10.4.11.12: icmp_req=2 ttl=64 time=0.476 ms
64 bytes from 10.4.11.12: icmp_req=3 ttl=64 time=0.244 ms
64 bytes from 10.4.11.12: icmp_req=4 ttl=64 time=0.362 ms
64 bytes from 10.4.11.12: icmp_req=5 ttl=64 time=0.263 ms
^C
--- 10.4.11.12 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4012ms
rtt min/avg/max/mdev = 0.244/0.435/0.832/0.215 ms
```



Y luego:

ping 10.4.11.11 (desde la pc2)

```
pc2
root@pc2:~# ping 10.4.11.11
PING 10.4.11.11 (10.4.11.11) 56(84) bytes of data.
64 bytes from 10.4.11.11: icmp_req=1 ttl=64 time=0.389 ms
64 bytes from 10.4.11.11: icmp_req=2 ttl=64 time=0.227 ms
64 bytes from 10.4.11.11: icmp_req=3 ttl=64 time=0.144 ms
64 bytes from 10.4.11.11: icmp_req=4 ttl=64 time=0.213 ms
64 bytes from 10.4.11.11: icmp_req=5 ttl=64 time=0.272 ms
^C
--- 10.4.11.11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3996ms
rtt min/avg/max/mdev = 0.144/0.249/0.389/0.081 ms
```

Obteniendo nuevamente un resultado exitoso y sacando como conclusión que las máquinas se pueden contactar entre sí.

#### 4. Cambie la configuración de los nombres de los equipos (temporal y permanente) asignándoles tyr11 y tyr12 a pc1 y pc2 respectivamente.

Para cambiar la configuración de los nombres de manera temporal, basta con utilizar el comando

hostname tyr11 (a la pc1)

y luego

hostname (para visualizar que el cambio se realizó de manera exitosa)

```
pc1
root@pc1:~# hostname tyr11
root@pc1:~# hostname
tyr11
root@pc1:~# █
```

hostname tyr12 (a la pc2)

y luego

hostname (para visualizar que el cambio se realizó de manera exitosa)

```
pc2
root@pc2:~# hostname tyr12
root@pc2:~# hostname
tyr12
root@pc2:~# █
```

Para hacer este cambio permanente abrimos un archivo de texto ubicado en “/etc/hostname” utilizamos nano lo editamos y lo guardamos.

La próxima vez que el sistema arranque, el nombre del equipo será el que editamos en el archivo de texto.

Para probar que el cambio anterior fue temporal, opté por reiniciar el laboratorio y luego verificar que el nombre de la pc no se había quedado modificado.

Efectivamente, al reiniciar la máquina, el nombre no se guardó.

```
pc1
GNU nano 2.2.6 File: hostname
pc1
```

Ahora, modifico dicho fichero para realizar el cambio de manera permanente.

```
pc1
GNU nano 2.2.6 File: hostname
tyr11
```

Y repito el procedimiento en la pc2:

```
pc2
GNU nano 2.2.6 File: hostname
tyr12
```

## 5) Resolución de nombres de hosts a direcciones IP.

### a. Configurar la resolución de nombres locales en ambos hosts con la información contenida en el punto 4.

Para cambiar la resolución de nombres locales, utilizo los siguientes comandos:

```
cd /etc
```

```
nano hosts
```

Luego, agrego la dirección ip de la pc2 y su nombre de host.

```
pc1
GNU nano 2.2.6 File: hosts
10.4.11.12 tyr12
127.0.0.1 localhost localhost.localdomain

::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

### b. Verificar que es posible contactar ambos equipos de la red utilizando nombres de host.

Ahora, realizo la prueba de ping utilizando el nombre de host en lugar de su dirección ip.

Los comandos utilizados son:

```
ping tyr12 (para pc1)
```

```
ping tyr11 (para pc2)
```



```
pc1
root@pc1:/etc# ping tyr12
PING tyr12 (10.4.11.12) 56(84) bytes of data.
64 bytes from tyr12 (10.4.11.12): icmp_req=1 ttl=64 time=0.308 ms
64 bytes from tyr12 (10.4.11.12): icmp_req=2 ttl=64 time=0.630 ms
64 bytes from tyr12 (10.4.11.12): icmp_req=3 ttl=64 time=0.704 ms
64 bytes from tyr12 (10.4.11.12): icmp_req=4 ttl=64 time=0.433 ms
^C
--- tyr12 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3024ms
rtt min/avg/max/mdev = 0.308/0.518/0.704/0.159 ms
root@pc1:/etc#
```

```
pc2
root@pc2:/# ping tyr11
PING tyr11 (10.4.11.11) 56(84) bytes of data.
64 bytes from tyr11 (10.4.11.11): icmp_req=1 ttl=64 time=0.708 ms
64 bytes from tyr11 (10.4.11.11): icmp_req=2 ttl=64 time=0.255 ms
64 bytes from tyr11 (10.4.11.11): icmp_req=3 ttl=64 time=0.432 ms
64 bytes from tyr11 (10.4.11.11): icmp_req=4 ttl=64 time=0.337 ms
^C
--- tyr11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3011ms
rtt min/avg/max/mdev = 0.255/0.433/0.708/0.170 ms
root@pc2:/#
```

## 6) Ver la tabla de ruteo definida en el equipo. ¿Cuáles son las redes accesibles?

Para ver la tabla de ruteo definida en el equipo, utilizo el comando:

```
ip route show
```

Obteniendo los siguientes resultados:

```
pc1
root@pc1:/etc# ip route show
10.4.11.0/24 dev eth0 proto kernel scope link src 10.4.11.11
root@pc1:/etc#
```

```
pc2
root@pc2:/# ip route show
10.4.11.0/24 dev eth0 proto kernel scope link src 10.4.11.12
root@pc2:/#
```

## 7. Agregar la dirección 10.4.11.30 como ruta por defecto para acceder a otras redes. Verificar nuevamente la tabla de ruteo.

Para agregar una ruta por defecto para acceder a otras redes, debo utilizar el comando:

```
ip route add default via 10.4.11.30
```

y luego, si ejecutamos el comando

```
ip route show
```

nos devolvería que esa interfaz tiene la ip 10.4.11.30 como vía por defecto.

```
pc1
root@pc1:/etc# ip route add default via 10.4.11.30
root@pc1:/etc# ip route show
default via 10.4.11.30 dev eth0
10.4.11.0/24 dev eth0 proto kernel scope link src 10.4.11.11
root@pc1:/etc#
```

**8. Realizar una captura de las PDU intercambiadas mientras se utiliza el comando ping para verificar conectividad con otro equipo. Las acciones que debe realizar son:**

**a. Iniciar la captura en una terminal del host anfitrión (utilizando el comando *vdump*), redireccionando la salida a un archivo para su posterior análisis.**

Para realizar la captura, utilizamos dentro de la terminal del host el comando:

```
vdump A > captura.pcap
```

```
juanma@ubuntu:~/Desktop$ vdump A > captura.pcap
Running ==> uml_dump A
```

Nota: Es importante ejecutar dicho comando en la terminal del host y no de las máquinas virtuales.

**b. En *pc1* ejecutar el comando ping para enviar a *pc2* exactamente 3 mensajes ICMP Echo Request (consulte el manual de ping).**

Una vez iniciada la captura, procedí a realizar un ping a la pc2 (desde pc1) obteniendo los siguientes resultados:

```
pc1
root@tyr11:~# ping 10.4.11.12 -c 3
PING 10.4.11.12 (10.4.11.12) 56(84) bytes of data.
64 bytes from 10.4.11.12: icmp_req=1 ttl=64 time=1.24 ms
64 bytes from 10.4.11.12: icmp_req=2 ttl=64 time=0.373 ms
64 bytes from 10.4.11.12: icmp_req=3 ttl=64 time=0.569 ms

--- 10.4.11.12 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 0.373/0.729/1.247/0.375 ms
```

Para realizar esto utilicé el siguiente comando:

```
ping 10.4.11.12 -c 3
```

Siendo 10.4.11.12 la dirección IP de la pc2.

El parámetro -c me permite indicar la cantidad de pings que quiero realizar (ya que, si no defino un valor para dicho parámetro, se ejecuta de manera indeterminada).

c. Una vez obtenida la respuesta del comando ping (deberán recibirse tres respuestas), detener la captura (finalizar el proceso vdump presionando Ctrl+C).

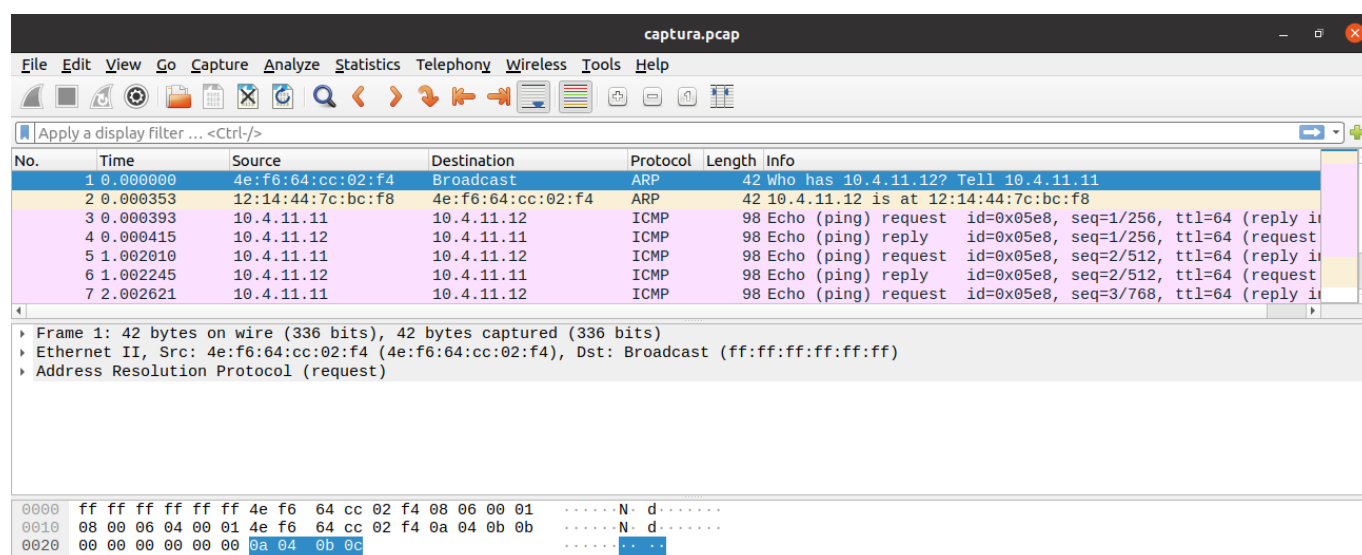
Para detener la captura debo posicionarme en la consola y presionar ctrl + c.

```
juanma@ubuntu:~/Desktop$ vdump A > captura.pcap
Running ==> uml_dump A
^CCaught signal 2, cleaning up and exiting
```

d. Analizar el volcado del programa de captura utilizando la aplicación wireshark (o cualquier otro analizador de tráfico que permita leer archivos en formato pcap), representando en un gráfico ideado por usted el intercambio de mensajes. Indicar cuál es la función de cada uno identificando los datos de encabezados más relevantes.

Para abrir el archivo que contiene la captura (llamado captura.pcap) con el analizador de tráfico llamado Wireshark debo posicionarme en la carpeta que contiene dicho archivo y ejecutar el comando:

```
wireshark captura.pcap
```



En este programa, tenemos varias secciones importantes.

La primera columna (No.) nos indica el número de paquete, el cual se encuentra clasificado por orden de llegada.

La segunda columna (Time) indica el tiempo que transcurrió entre un paquete y otro.

La tercera columna (Source) nos indica quien generó el paquete, o sea, el origen del paquete (indica la ip, Mac o hostname de la máquina que envió el paquete).

La cuarta columna (Destination) indica el destino del paquete, es decir, para quien está dirigido.

La quinta columna (Protocol), como lo indica su nombre, indica el protocolo al cual está asociado el paquete (en este caso, los pings están asociados al protocolo ICMP).

La sexta columna (Lenght) indica la longitud del paquete.

La última columna (Info) nos proporciona información adicional acerca de los paquetes.

Una vez que ya sabemos interpretar cada columna, podemos ver el detalle de cada paquete en la siguiente sección:

Una vez ampliados los elementos, podemos sacar datos importantes como por ejemplo a que capa del modelo OSI está asociado.

En este caso, la primera parte está asociada a la capa física.

```
▼ Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
  Encapsulation type: Ethernet (1)
  Arrival Time: Apr  9, 2021 19:50:52.084705000 PDT
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1618023052.084705000 seconds
  [Time delta from previous captured frame: 0.000040000 seconds]
  [Time delta from previous displayed frame: 0.000040000 seconds]
  [Time since reference or first frame: 0.000393000 seconds]
  Frame Number: 3
```

La segunda parte, referida a la capa de enlace, destaca un dato importante conocido como dirección MAC, la cual es un identificador compuesto de 6 bloques de 2 caracteres hexadecimales, que corresponden a una tarjeta de red y es único.

```
▼ Ethernet II, Src: 4e:f6:64:cc:02:f4 (4e:f6:64:cc:02:f4), Dst: 12:14:44:7c:bc:f8 (12:14:44:7c:bc:f8)
  ▸ Destination: 12:14:44:7c:bc:f8 (12:14:44:7c:bc:f8)
  ▸ Source: 4e:f6:64:cc:02:f4 (4e:f6:64:cc:02:f4)
  Type: IPv4 (0x0800)
```

La tercera parte, está asociada a la capa 3 del modelo OSI, es decir a la capa de red. En esta capa destaco al protocolo IP, en este caso, en la versión 4 (IPv4).

Como ya se ha indicado Src hace referencia al emisor del “mensaje” y Dst hace referencia al destinatario, es decir quien va a recibir ese mensaje.

```
▼ Internet Protocol Version 4, Src: 10.4.11.11, Dst: 10.4.11.12
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 84
  Identification: 0xb774 (46964)
  ▸ Flags: 0x4000, Don't fragment
  Fragment offset: 0
  Time to live: 64
```

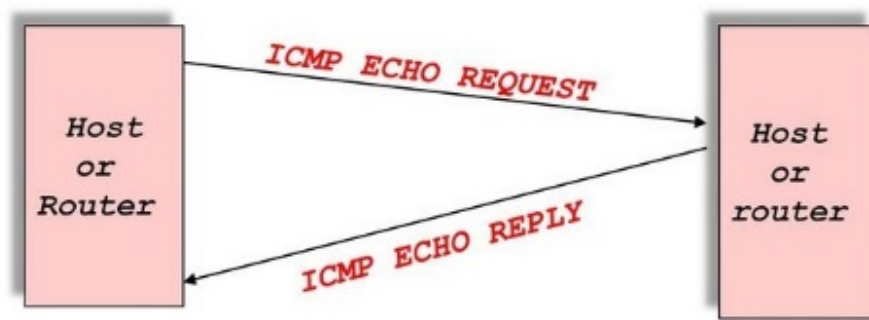
La última parte, nos proporciona información acerca del tipo de ICMP.

```
▼ Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x1d74 [correct]
  [Checksum Status: Good]
  Identifier (BE): 1512 (0x05e8)
  Identifier (LE): 59397 (0xe805)
  Sequence number (BE): 1 (0x0001)
  Sequence number (LE): 256 (0x0100)
```

En el caso de los pings, si fue solicitud (request) o respuesta (reply).

El checksum, que nos ayuda a saber si hubo errores en la transmisión de datos (verifica la integridad).

El funcionamiento del Ping se explica en el siguiente gráfico:



Suponiendo que el primer host es pc1, y el segundo pc2, pc1 (en este caso, source) envía una solicitud (request) a pc2 (destination) y en caso de poder comunicarse, pc2 le envía la respuesta (reply).