NETWORKING LINUX:

Introducción

Así como el tema de las redes es vasto, también lo es la cantidad de comandos que se pueden usar para configurarlo y controlarlo. Centraremos nuestra atención en solo algunos de los más utilizados. Los comandos elegidos para el examen incluyen los que se usan para monitorear redes y los que se usan para transferir archivos. Además, vamos a explorar el programa ssh que se utiliza para realizar inicios de sesión remotos. Este capítulo cubrirá los siguientes comandos:

ping : envía un ICMP ECHO_REQUEST a los hosts de la red

traceroute : imprime el seguimiento de los paquetes de ruta en un host de red

ip : muestra / manipula el enrutamiento, los dispositivos, el enrutamiento de políticas y los túneles

netstat : imprime conexiones de red, tablas de enrutamiento, estadísticas de inferfaces, conexiones de enmascaramiento y membresías de multidifusión

ftp : programa de transferencia de archivos de Internet

wget : descargador de red no interactivo

ssh : cliente OpenSSH SSH (programa de inicio de sesión remoto)

Debemos estar familiarizados con los siguientes términos:

1. Dirección de Protocolo de Internet (IP)
2. Nombre de dominio y host
3. Identificador uniforme de recursos (URI)

Examinar y monitorear una red

Incluso si no es el administrador del sistema, a menudo resulta útil examinar el rendimiento y el funcionamiento de una red.

ping

El comando de red más básico es ping. El comando ping envía un paquete de red especial llamado ICMP ECHO_REQUEST a un host especificado. La mayoría de los dispositivos de red que reciben este paquete le responderán, lo que permitirá verificar la conexión de red.

Es posible configurar la mayoría de los dispositivos de red (incluidos los hosts de Linux) para ignorar estos paquetes. Esto generalmente se hace por razones de seguridad para ocultar parcialmente un host de un atacante potencial. También es común que los firewalls se configuren para bloquear el tráfico ICMP.

Por ejemplo, para ver si podemos llegar a linuxcommand.org (uno de nuestros sitios favoritos), podemos usar ping así:

```
[~]$ ping linuxcommand.org
```

Una vez iniciado, ping continúa enviando paquetes en un intervalo especificado (el valor predeterminado es un segundo) hasta que se interrumpe.

Después de que se interrumpa (en este caso después del sexto paquete) presionando CTRL-C imprime estadísticas de rendimiento. Una red que funcione correctamente presentará una pérdida de paquetes del 0 por ciento. Un 'ping' exitoso indicará que los elementos de la red (sus tarjetas de interfaz, cableado, enrutamiento y puertas de enlace) están en general en buen estado de funcionamiento.

traceroute

El programa traceroute (algunos sistemas usan el programa tracepath similar en su lugar) enumera todos los 'saltos' que el tráfico de red necesita para llegar del sistema local a un host específico. Por ejemplo, para ver la ruta tomada para llegar a slashdot.org, haríamos esto:

```
[~]$ traceroute slashdot.org
```

La salida se ve así:

```
[~]$ traceroute slashdot.org
traceroute to slashdot.org (216.105.38.15), 64 hops max
  1   192.168.1.1  0,554ms  0,461ms  0,555ms
  2   *  *  *
  3   *  *  *
  4   81.41.250.110  29,107ms  29,458ms  29,551ms
  5   80.58.96.117  28,411ms  27,795ms  28,949ms
  6   213.140.51.56  26,434ms  28,948ms  33,343ms
  7   213.140.33.89  59,887ms  60,798ms  59,397ms
  8   213.140.35.238  114,437ms  113,873ms  113,662ms
  9   94.142.117.55  138,925ms  138,434ms  138,145ms
 10   94.142.107.159  142,002ms  141,297ms  141,214ms
 11   94.142.107.159  147,178ms  139,103ms  139,183ms
 12   *  *  *
 13   63.146.26.137  141,052ms  150,439ms  140,354ms
 14   63.146.26.137  144,076ms  143,886ms  143,696ms
 15   65.126.18.126  194,092ms  193,198ms  193,271ms
 16   207.158.62.109  197,421ms  196,305ms  197,311ms
 17   209.216.192.66  187,063ms  186,876ms  186,695ms
 18   216.105.38.15  194,394ms  193,754ms  194,195ms
 19   *  *  *
 20   *  *  *
 21   *  *  *
 22   *  *  *
```

En el resultado, podemos ver que conectarse desde nuestro sistema de prueba a slashdot.org requiere atravesar 22 enrutadores. Para los enrutadores que proporcionaron información de identificación, vemos sus nombres de host, direcciones IP y datos de rendimiento, que incluyen tres muestras de tiempo de ida y vuelta desde el sistema local al enrutador. Para los enrutadores que no proporcionan información de identificación (debido a la configuración del enrutador, congestión de la red, cortafuegos, etc.), vemos asteriscos como en la línea del salto número 2. En los casos en que la información de enrutamiento está bloqueada, a veces podemos superar esto agregando ya sea la opción -T o -I del comando traceroute.

ip

El programa ip es una herramienta de configuración de red multipropósito que hace uso de la gama completa de funciones de red disponibles en los núcleos Linux modernos. Reemplaza el programa ifconfig anterior y ahora obsoleto. Con ip, podemos examinar las interfaces de red y la tabla de enrutamiento de un sistema.

```
[~]$ 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: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether d4:c9:ef<img src="https://github.global.ssl.fastly.net/images/icons/emoji/de.png?v5" alt=":de:
    inet 192.168.1.38/24 brd 192.168.1.255 scope global dynamic noprefixroute eno1
       valid_lft 29202sec preferred_lft 29202sec
    inet6 fe80::6d2d:1ecd:7ac1:41d/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1
    link/none
    inet 10.8.0.1/24 brd 10.8.0.255 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::e7d0:2289:fcd3:ee06/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
4: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN group default qlen 1000
    link/none
    inet 192.168.6.2/24 scope global wg0
       valid_lft forever preferred_lft forever
5: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:fa:9c:41 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
6: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:fa:9c:41 brd ff:ff:ff:ff:ff:ff
7: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:1a:eb:48:3b 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
```

En el ejemplo anterior, vemos que nuestro sistema de prueba tiene dos interfaces de red. La primera, llamada lo, es la interfaz de bucle invertido, una interfaz virtual que el sistema usa para 'hablar consigo mismo', y la segunda, llamada eth0, es la interfaz Ethernet.

Al realizar diagnósticos de red casuales, lo importante que debe buscar son la presencia de la palabra UP en la primera línea para cada interfaz, lo que indica que la interfaz de red está habilitada, y la presencia de una dirección IP válida en el campo inet en la tercera. línea. Para los sistemas que utilizan el Protocolo de configuración dinámica de host (DHCP), una dirección IP válida en este campo verificará que el DHCP esté funcionando.

netstat

El programa netstat se utiliza para examinar varias configuraciones y estadísticas de red. Mediante el uso de sus muchas opciones, podemos ver una variedad de características en nuestra configuración de red. Usando la opción -ie, podemos examinar las interfaces de red en nuestro sistema.

`[~]$ netstat -ie`

```
Tabla de la interfaz del núcleo
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:1a:eb:48:3b  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

```
        inet 192.168.1.38  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::6d2d:1ecd:7ac1:41d  prefixlen 64  scopeid 0x20<link>
        ether d4:c9:ef<img src="https://github.global.ssl.fastly.net/images/icons/emoji/de.png?v5" alt=":de:"
        RX packets 1973220  bytes 2860471957 (2.8 GB)
        RX errors 0  dropped 121  overruns 0  frame 0
        TX packets 546589  bytes 50587231 (50.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 20  memory 0xf7c00000-f7c20000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Bucle local)
        RX packets 5879  bytes 1717165 (1.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5879  bytes 1717165 (1.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.8.0.1  netmask 255.255.255.0  destination 10.8.0.1
        inet6 fe80::e7d0:2289:fcd3:ee06  prefixlen 64  scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 100  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 486  bytes 93104 (93.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:fa:9c:41  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wg0: flags=209<UP,POINTOPOINT,RUNNING,NOARP>  mtu 1420
        inet 192.168.6.2  netmask 255.255.255.0  destination 192.168.6.2
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 122  dropped 0 overruns 0  carrier 0  collisions 0
```

El uso de la opción -r mostrará la tabla de enrutamiento de red del kernel.

Esto muestra cómo la red está configurada para enviar paquetes de una red a otra:

```
[~]$ netstat -r
Tabla de rutas IP del núcleo
Destino         Pasarela        Genmask         Indic   MSS Ventana irtt Interfaz
default         _gateway        0.0.0.0         UG        0 0          0 eno1
10.8.0.0        0.0.0.0         255.255.255.0   U         0 0          0 tun0
link-local      0.0.0.0         255.255.0.0     U         0 0          0 eno1
172.17.0.0      0.0.0.0         255.255.0.0     U         0 0          0 docker0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eno1
192.168.6.0     0.0.0.0         255.255.255.0   U         0 0          0 wg0
192.168.122.0   0.0.0.0         255.255.255.0   U         0 0          0 virbr0
```

En este ejemplo simple, vemos una tabla de enrutamiento típica para una máquina cliente en una red de área local (LAN) detrás de un firewall / enrutador. La primera línea de la lista muestra el destino 192.168.1.0. Las direcciones IP que terminan en cero se refieren a redes en lugar de hosts individuales, por lo que este destino significa cualquier host en la LAN. El siguiente campo, Puerta de enlace, es el nombre o la dirección IP de la puerta de enlace (enrutador) que se utiliza para ir del host actual a la red de destino. Un asterisco en este campo indica que no se necesita puerta de enlace.

La última línea contiene el destino predeterminado. Esto significa cualquier tráfico destinado a una red que no figura en la tabla. En nuestro ejemplo, vemos que la puerta de enlace se define como un enrutador con la dirección 192.168.1.1, que presumiblemente sabe qué hacer con el tráfico de destino.

Al igual que ip, el programa netstat tiene muchas opciones, y solo hemos analizado un par. Consulte las páginas de manual de ip y netstat para obtener una lista completa.

Transporte de archivos a través de una red

¿De qué sirve una red a menos que podamos mover archivos a través de ella? Hay muchos programas que transfieren datos a través de redes. Cubriremos dos de ellos ahora y varios más en secciones posteriores.

tp

Uno de los verdaderos programas 'clásicos', ftp recibe su nombre del protocolo que utiliza, el Protocolo de transferencia de archivos. FTP fue una vez el método más utilizado para descargar archivos a través de Internet. La mayoría, si no todos, los navegadores web lo admiten, y a menudo ve URI que comienzan con el protocolo ftp: //.

Antes de que existieran los navegadores web, existía el programa ftp. ftp se utiliza para comunicarse con servidores FTP, máquinas que contienen archivos que se pueden cargar y descargar a través de una red.

FTP (en su forma original) no es seguro porque envía nombres de cuentas y contraseñas en texto sin cifrar. Esto significa que no están encriptados y cualquiera que esté olfateando la red puede verlos. Debido a esto, casi todo el FTP que se realiza a través de Internet se realiza mediante servidores FTP anónimos. Un servidor anónimo permite que cualquiera pueda iniciar sesión con el nombre de inicio de sesión 'anónimo' y una contraseña sin sentido.

En el ejemplo que sigue, mostramos una sesión típica con el programa ftp descargando una imagen iso de Ubuntu ubicada en el directorio /pub/cd_images/ubuntu-18.04 del servidor de archivos del servidor FTP anónimo:

```
[]~]$ ftp fileserver
Connected to fileserver.localdomain.
220 (vsFTPd 2.0.1)
Name (fileserver:me): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd pub/cd_images/ubuntu-18.04
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-rw-r--
1 500
500
733079552 Apr 25 03:53 ubuntu-18.04-desktop-amd64.iso
226 Directory send OK.
ftp> lcd Desktop
Local directory now /home/me/Desktop
ftp> get ubuntu-18.04-desktop-amd64.iso
local: ubuntu-18.04-desktop-amd64.iso remote: ubuntu-18.04-desktop-amd64.iso
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for ubuntu-18.04-desktop-amd64.iso (733079552 bytes).
226 File send OK.
733079552 bytes received in 68.56 secs (10441.5 kB/s)
ftp> bye
```

Explicación de algunos comandos utilizados en el ftp:

Comando:
Sentido

ftp fileserver :

Invoque el programa ftp y haga que se conecte al servidor de archivos del servidor FTP.

anonymous :

Nombre de inicio de sesión. Después de la solicitud de inicio de sesión, aparecerá una solicitud de contraseña. Algunos servidores aceptarán una contraseña en blanco; otros requerirán una contraseña en forma de dirección de correo electrónico. En ese caso, intente algo como user@example.com.

cd pub/cd_images/ubuntu-18.04 :

Cambie al directorio del sistema remoto que contiene el archivo deseado. Tenga en cuenta que en la mayoría de los servidores FTP anónimos, los archivos para descarga pública se encuentran en algún lugar del directorio pub.

ls :

Enumere el directorio en el sistema remoto.

lcd Desktop:

Cambie el directorio en el sistema local a ~/ Desktop. En el ejemplo, se invocó el programa ftp cuando el directorio de trabajo era ~. Este comando cambia el directorio de trabajo a ~/Desktop.

get ubuntu-18.04-desktop-amd64.iso :

Dígale al sistema remoto que transfiera el archivo ubuntu-18.04-desktop-amd64.iso al sistema local.
Dado que el directorio de trabajo en el sistema local se cambió a ~/Desktop, el archivo se descargará allí.

bye :

Cierre la sesión del servidor remoto y finalice la sesión del programa ftp. También se pueden utilizar los comandos quit y exit.

Si escribe help en el indicador ftp>, se mostrará una lista de los comandos admitidos. Al usar ftp en un servidor donde se han otorgado permisos suficientes, es posible realizar muchas tareas ordinarias de administración de archivos. Es torpe, pero funciona.

lftp --- un ftp mejor

ftp no es el único cliente FTP de línea de comandos. De hecho, hay muchos. Uno de los mejores (y más populares) es lftp de Alexander Lukyanov. Funciona de manera muy similar al programa ftp tradicional, pero tiene muchas características de conveniencia adicionales que incluyen compatibilidad con múltiples protocolos (incluido HTTP), reintento automático en descargas fallidas, procesos en segundo plano, finalización de pestañas de nombres de rutas y muchas más.

wget

Otro programa de línea de comandos popular para la descarga de archivos es wget. Es útil para descargar contenido de sitios web y FTP. Se pueden descargar archivos individuales, múltiples archivos e incluso sitios completos. Para descargar la primera página de linuxcommand.org, podríamos hacer esto:

```
[~]$ wget http://linuxcommand.org/index.php
```

Las muchas opciones del programa permiten a wget descargar de forma recursiva, descargar archivos en segundo plano (lo que le permite cerrar sesión pero continuar con la descarga) y completar la descarga de un archivo parcialmente descargado. Estas características están bien documentadas en su página de manual mejor que el promedio.

Comunicación segura con hosts remotos

Durante muchos años, los sistemas operativos similares a Unix han tenido la capacidad de administrarse de forma remota a través de una red. En los primeros días, antes de la adopción generalizada de Internet, se utilizaban un par de programas populares para iniciar sesión en hosts remotos. Estos fueron los programas rlogin y telnet. Estos programas, sin embargo, adolecen del mismo defecto fatal que el programa ftp; transmiten todas sus comunicaciones (incluidos los nombres de inicio de sesión y las contraseñas) en texto sin cifrar. Esto los hace totalmente inapropiados para su uso en la era de Internet.

ssh

Para abordar este problema, se desarrolló un nuevo protocolo llamado Secure Shell (SSH). SSH resuelve los dos problemas básicos de la comunicación segura con un host remoto.

1. Autentica que el host remoto es quien dice ser (evitando así los llamados ataques man-in-the-middle).

2. Cifra todas las comunicaciones entre los hosts locales y remotos.

SSH consta de dos partes. Un servidor SSH se ejecuta en el host remoto, escuchando las conexiones entrantes, de forma predeterminada, en el puerto 22, mientras que un cliente SSH se usa en el sistema local para comunicarse con el servidor remoto.

La mayoría de las distribuciones de Linux incluyen una implementación de SSH llamada OpenSSH del proyecto OpenBSD. Algunas distribuciones incluyen tanto los paquetes de cliente como de servidor de forma predeterminada (por ejemplo, Red Hat), mientras que otras (como Ubuntu) proporcionan solo el cliente. Para permitir que un sistema reciba conexiones remotas, debe tener el paquete de servidor OpenSSH instalado, configurado y en ejecución, y (si el sistema está funcionando o está detrás de un firewall) debe permitir conexiones de red entrantes en el puerto TCP 22.

Si no tiene un sistema remoto al que conectarse pero desea probar estos ejemplos, asegúrese de que el paquete del servidor OpenSSH-server esté instalado en su sistema y use localhost como el nombre del host remoto. De esa manera, su máquina creará conexiones de red consigo misma.

El programa cliente SSH que se utiliza para conectarse a servidores SSH remotos se llama, apropiadamente, ssh. Para conectarse a un host remoto llamado remote-sys, usaríamos el programa cliente ssh así:

como abilitar el servidor ssh en Ubuntu 20.04

1. Ctrl + Alt + T
2. install OpenSSH : sudo apt-get install openssh-server
3. confirmar la instalacion
4. está activado ssh. comprobarlo con el comando: sudo service ssh status.

```
[~]$ sudo apt-get install openssh-server
[~]$ sudo service ssh status
```

Cómo conectar un servidor remoto usando SSH en Ubuntu 20.04

Para conectar el servidor Ubuntu remoto a través de OpenSSH, debe conocer la dirección IP o el nombre de dominio / computadora de ese servidor o escritorio en particular. Por ejemplo, quiero conectar un servidor disponible en otro edificio de mi

oficina, por lo tanto, primero le preguntaré a alguien o averiguaré manualmente la dirección IP del servidor. El comando para ifconfig en Linux e ipconfig para Windows.

comprobar el estado del servicio ssh:

Once openssh-server package is installed in your Server, you can now see ssh service available under /etc/init.d directory. Now you can check the status of ssh service by using /etc/init.d/ssh status command as shown below.

```
[~]$ /etc/init.d/ssh status
```

```
[~]$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:M1ksXl2Qgtlcidp8+GW2lSxK1w8NVJLflNS4A1kqh+o.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
hernani@localhost's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 device has a firmware upgrade available.
Run `fwupdmgr get-upgrades` for more information.


38 actualizaciones se pueden instalar inmediatamente.
7 de estas actualizaciones son una actualización de seguridad.
Para ver estas actualizaciones adicionales ejecute: apt list --upgradable


7 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log
Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Nov  6 23:33:03 2020
```

Inicie sesión en Ubuntu 20.04 usando el puerto ssh 22

Ahora, si intenta conectar su servidor Ubuntu 20.04 usando el puerto ssh 22, entonces debería poder conectarse como se muestra a continuación.

En caso de que aún no pueda conectarse, le sugiero que verifique y realice los pasos a continuación:

a) Verifique su archivo /etc/ssh/sshd_config y asegúrese de que la línea del puerto 22 no esté comentada allí. Si no está descomentado, puede descomentarlo y reiniciar el servicio ssh usando systemctl restart ssh o usando el comando /etc/init.d/ssh restart.

b) Asegúrese de que ssh esté permitido desde el firewall ufw si está ejecutando este firewall. Si no está permitido, debe permitirlo y luego volver a intentar ssh el servidor Ubuntu 20.04. Deberia de funcionar.

```
[~]$ sudo apt upgrade
```

Ahora usa la siguiente estructura de comando

```
[~]$ ssh username@ipaddress
```

como comprobar la usual route usando ip commando

del paquete net-tools, usamos 'route -n' comprobamos todas las rutas estatias:

```
[~]$ route -n
Tabla de rutas IP del núcleo
Destino         Pasarela        Genmask         Indic Métric Ref    Uso Interfaz
0.0.0.0         192.168.1.1     0.0.0.0         UG    100    0        0 eno1
10.8.0.0        0.0.0.0         255.255.255.0   U     0      0        0 tun0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eno1
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 eno1
192.168.6.0     0.0.0.0         255.255.255.0   U     0      0        0 wg0
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```

scp y sftp

scp

El paquete OpenSSH también incluye dos programas que pueden hacer uso de un túnel encriptado SSH para copiar archivos a través de la red. El primero, scp (copia segura), se usa de manera muy similar al programa cp familiar para copiar archivos. La diferencia más notable es que los nombres de ruta de origen o destino pueden estar precedidos por el nombre de un host remoto, seguido de dos puntos. Por ejemplo, si quisiéramos copiar un documento llamado document.txt de nuestro directorio de inicio en el sistema remoto, remote-sys, al directorio de trabajo actual en nuestro sistema local, podríamos hacer esto:

`[~]$ scp remote-sys:document.txt`

Al igual que con ssh, puede aplicar un nombre de usuario al principio del nombre del host remoto si el nombre de la cuenta del host remoto deseado no coincide con el del sistema local.

`[~]$ scp bob@remote-sys:document.txt`

sftp

El segundo programa de copia de archivos SSH es sftp, que, como su nombre lo indica, es un reemplazo seguro del programa ftp.

sftp funciona de manera muy similar al programa ftp original que usamos anteriormente; sin embargo, en lugar de transmitir todo en texto plano, utiliza un túnel encriptado SSH.

sftp tiene una ventaja importante sobre el ftp convencional en el sentido de que no requiere que se ejecute un servidor FTP en el host remoto. Solo requiere el servidor SSH. Esto significa que cualquier máquina remota que pueda conectarse con el cliente SSH también puede usarse como un servidor similar a FTP. Aquí hay una sesión de muestra.

```
[~]$ sftp remote-sys
Connecting to remote-sys...
me@remote-sys's password:
sftp> ls
ubuntu-8.04-desktop-i386.iso
sftp> lcd Desktop
sftp> get ubuntu-8.04-desktop-i386.iso
```

```
Fetching /home/me/ubuntu-8.04-desktop-i386.iso to ubuntu-8.04-desktop-i386.iso
/home/me/ubuntu-8.04-desktop-i386.iso 100% 699MB
7.4MB/s
01:35
sftp> bye
```

El protocolo SFTP es compatible con muchos de los administradores de archivos gráficos que se encuentran en las distribuciones de Linux. Usando GNOME o KDE, podemos ingresar un URI que comience con sftp: // en la barra de ubicación y operar en archivos almacenados en un sistema remoto que ejecuta un servidor SSH. """