3. Networking

Se ha preparado un laboratorio con dos máquinas virtuales, una con Ubuntu y otra con CentOS. Ambas tienen dos interfaces de red, una que da conectividad al exterior y otra red interna. El entorno será accesible en el directorio **vagrant_net**.

1. Direccionamiento IP

a. Necesitamos montar una red local. Esta red local estará segmentada de la siguiente manera: un segmento donde se ubicaran todos los webservers. Se estima que con el crecimiento que tendrá la empresa tendremos unos 300. Otro segmento donde irán servidores de aplicación, en ese caso la estimación es de unos 100. Por último necesitaremos un segmento donde ubicar las BBDD. No habrá más de 10 BBDD en total.
Proponer un direccionamiento adecuado y justificarlo.

Nota: Se puede usar el ipcalc como ayuda.

webservers 172.168.1.0/23
appsservers 192.168.1.0/25
bbdd 192.168.2.0/28

2. Subneting

a. Dada la siguiente dirección IPv4 172.16.68.230, con una máscara de red original de 255.255.0.0 y la nueva subred sea 255.255.240.0. Determinar los siguientes parámetros:
   i. Número de bits de la subred
   ii. Número de subredes que se van a crear
   iii. Número de bits de hosts
   iv. Número de hosts por subred
   v. Dirección de red de la IP dada
   vi. Primer host en la red de la IP dada
   vii. Último host de la red de la IP dada
   viii. Dirección de broadcast de la red

Nota: Se puede usar el ipcalc como ayuda.

Solución: https://www.ipchalk.com/how-to/ipv4-subnet-by-mask/
https://www.binaryhexconverter.com/binary-to-decimal-converter

3. Verificar configuración IP

a. Open a root shell.
b. Type **ip -s link**. This shows all existing network connections, in addition to statistics about the number of packets that have been sent and associated error messages.
c. Type **ip addr show**. You'll see the current address assignments for network interfaces on your server.

4. Verificar conectividad de red

a. Open a root shell to your server and type **ip addr show**. This shows the current network configuration. Note the IPv4 address that is used. Notice the network device names that are used; you need these later in this exercise.
b. Type **ip route show** to verify routing configuration.
c. If your computer is connected to the Internet, you can now use the **ping** command to verify the connection to the Internet is working properly. Type **ping -c 4 8.8.8.8**, for instance, to send four packets to IP address 8.8.8.8. If your Internet connection is up and running, you should get "echo reply" answers.
d. Type **ip addr add 10.0.0.10/24 dev <yourdevicename>**.
e. Type **ip addr show**. You'll see the newly set IP address, in addition to the IP address that was already in use.
f. Type **ifconfig**. Notice that you do not see the newly set IP address (and there are no options with the **ifconfig** command that allow you to see it). This is one example why you should not use the **ifconfig** command anymore.
g. Type **ss -tul**. You'll now see a list of all UDP and TCP ports that are listening on your server.

5. Realizar consultas DNS

a. Abrir una shell y ejecutar **dig ubuntu.com** obtendremos la lista de direcciones que corresponden a ese dominio en la sección de respuesta. Al haber mas de uno, cuando nos conectemos a ubuntu.com se eligirá nua al azar.
b. Para ver una respuesta más simplificada podemos ejecutar la misma consulta de la siguiente manera: **dig +short ubuntu.com**
c. Podemos consultar a través de otro servidor DNS, si pensamos que el que tenemos por defecto no está funcionando como deberia: **dig +short ubuntu.com @8.8.8.8**
d. Podemos consultar los servidores de correo que tiene configurados el dominio adjuntando el tipo de registro que estamos consultando: **dig ubuntu.com MX**
e. Podemos desde una IP hacer una consulta del nombre DNS con **dig -x 91.189.89.110**

f. Se puede consultar también todos los tiops de registros asociados a un domniio: **dig ubuntu.com ANY**

6. Configuración de la red por línea de comandos

a. Conectarse a la máquina ubuntu y cambiarse a root.
b. Verificamos las interfaces de red que tenemos configuradas con **ip addr show**
c. Eliminamos la interfaz de red que tenemos configurada con **ip addr del 3.3.3.3/24 dev enp0s8**
d. Verificamos de nuevo que la IP ya se ha desconfigurado
e. Configuraremos una nueva IP en otro segmento de red sobre la misma interfaz ejecutando **ip addr add 192.168.100.1/24 dev enp0s8**.
f. Verificamos que la IP 192.168.100.1 se ha configurado correctamente.
g. Reiniciamos la máquina y verificamos si mantiene esa IP. ¿qué ha pasado?

7. Configuración de la red persistente en Ubuntu

a. Conectarse a la máquina ubuntu y cambiarse a root.
b. Verificamos las interfaces de red que tenemos configuradas con **ip addr show**.
c. Anotamos el nombre de la interfaz de red que actualmente tiene la IP 3.3.3.3 asignada (debería ser enp0s8).
d. Editamos el fichero **/etc/netplan/50-vagrant.yaml** dejándolo de la siguiente manera:

--> 192.168.100.2/24

e. Reiniciamos el servicio de red **netplan try** y verificamos si la IP de la interfaz ha cambiado **ip add show enp0s8** y aplicamos con **netplan apply**
f. Reiniciamos la máquina y verificamos si se mantiene la IP.
g. ¿Cómo se configura la interfaz de red principal en vagrant? (enp0s3)

8. Configuración de la red persistente en Centos

a. Conectarse a la máquina centos y cambiarse a root.
b. Verificamos las interfaces de red que tenemos configuradas con **ip addr show**.
c. Anotamos el nombre de la interfaz de red que actualmente tiene la IP 2.2.2.2 asignada (debería ser eth1).
d. Editamos el fichero /etc/sysconfig/network-scripts/ifcfg-eth1 dejándolo de la siguiente manera:

NM_CONTROLLED=yes
BOOTPROTO=none

ONBOOT=yes
IPADDR=192.168.100.3
NETMASK=255.255.255.0
DEVICE=eth1
PEERDNS=no

e. Reiniciamos el servicio de red **ifdown eth1 && ifup eth1** y verificamos si la IP de la interfaz ha cambiado **ip add show eth1**
f. Reiniciamos la máquina y verificamos si se mantiene la IP.
g. ¿Cómo se configura la interfaz de red principal en vagrant? (eth0)

9. Uso de nmcli

a. Nos conectaremos a la máquina con CentOS y nos cambiamos a root para poder realizar modificaciones en la configuración de red.
b. Con **nmcli** verificamos la configuración de red de los dispositivos.
c. Con **nmcli device status** tendremos una salida tabular de las interfaces y veremos su estado.
d. Con **nmcli con show** obtenderemos una lista de los perfiles de conexión creados.
e. Para ver el detalle de un perfil en concreto usaremos: **nmcli connection show System\ eth1**
f. Si queremos obtener solo unos parámetros en concreto usaremos: **nmcli -g ip4.address,ip4.dns connection show System\ eth0**
g. Para modificar algún parámetro podemos pasarlo directamente desde la línea de comandos: **nmcli con mod System\ eth1 ipv4.method manual ipv4.addresses 192.168.100.6/24 ipv4.dns 8.8.8.8 ipv4.gateway 192.168.100.1**
h. Para activar la nueva configuración debemos bajar la interfaz y volverla a levantar, para ello usaremos:
   nmcli con down System\ eth1
   nmcli con up System\ eth1
i. Podemos verificar estos parámetros que hemos cambiado: **nmcli -g ip4.address,ip4.dns,ipv4.gateway connection show System\ eth1**
j. Vamos a crear un nuevo perfil: **nmcli connection add con-name Custom-eth1 ifname eth1 type ethernet ip4 192.168.200.6/24**
k. Como vemos con **nmcli connection show** este perfil nuevo se ha creado, pero no está activo. Para activarlo, bajaremos primero la conexión que está actualmente usando la interfaz y activaremos el nuevo perfil:
   nmcli con down System\ eth1
   nmcli con up Custom-eth1
l. Verificaremos los cambios con:
   nmcli con show
   nmcli
m. Los cambios en el perfil se pueden hacer también en modo edición: **nmcli con edit Custom-eth1** (se puede usar el tabulador para autocompletar)

n. Podemos verificar algún parámetro:
   nmcli> print ipv4.addresses
o. Y también se pueden modificar:
   nmcli> set ipv4.addresses 192.168.202.6/24
p. Deberemos guardar antes de salir para poder aplicar luego los cambios:
   nmcli> save
   nmcli> exit

10. Montar servidor DHCP

a. En la máquina con ubuntu instalaremos el paquete isc-dhcp-server con el comando **sudo apt-get install isc-dhcp-server**.
b. Configuraremos la interfaz de red secundaria con la IP 192.168.100.1: **ip addr add 192.168.100.1/24 dev enp0s8**
c. Editaremos el fichero **/etc/dhcp/dhcpd.conf** añadiendo el siguiente contenido:

```
subnet 192.168.100.0 netmask 255.255.255.0 {
        option routers 192.168.100.1 ;
        range 192.168.100.200 192.168.100.202 ;
        }
```

d. Especificaremos en el fichero **/etc/default/isc-dhcp-server** el nombre de la interfaz de red secundaria.
e. Reiniciaremos el servicio de dhcp **systemctl restart isc-dhcp-server.service**
f. Dejaremos un tail al syslog para verificar si nos están llegando peticiones al servidor DHCP: **tail -f /var/log/syslog | egrep -v "sda|sdb"**
g. En ese momento ya podemos arrancar la máquina con CentOS
h. Crearemos un nuevo perfil de red con: **nmcli connection add con-name dhcp-eth1 ifname eth1 type ethernet**
i. Y lo activaremos:
   nmcli con dow Custom-eth1
   nmcli con up dhcp-eth1
j. Una vez aplicado el nuevo perfil, la interfaz eth1 deberá obtener una IP por DHCP en el rango 192.168.100.200-192.168.100.202, lo veremos con **ip addr show**.
k. Verificar con un ping a la IP 192.168.100.1 si tenemos conectividad en ese segmento.

11. Verificar sincronización NTP

a. En las últimas versiones de las distribuciones el servicio de hora se gestiona a través del demonio Chronyd. Con el comando **timedatectl** podemos verificar si esta el reloj del sistema sincronizado con los relojes de internet
b. Para habilitar la sincronización con los servidores de internet (si no lo estaba ya) ejecutando **sudo timedatectl set-ntp on**.

c. Si queremos volver a usar el demonio ntpd deshabilitaremos el timesyncd con **sudo timedatectl set-ntp no**.
d. Instalaremos el ntp con **sudo apt-get install ntp**.
e. Para verificar la sincronización ntp ejecutaremos **sudo ntpq -p**, la línea que empiece con * indicará el servidor con el que se está sincronizando el servidor.

12. Crear mecanismo de autenticación RSA entre dos servidores

a. Conectarse al servidor ubuntu.
b. Ejecutar **ssh-keygen**. Cuando se nos pida por una passphrase, pulsar **Enter** para usar una configuración sin passphrase.
c. Cuando nos pregunte por la ruta donde almacenar nuestra clave privada, aceptaremos el nombre por defecto: ~/.ssh/id_rsa.
d. La clave privada está ahora en el fichero ~/.ssh/id_rsa file y la pública en el fichero ~/.ssh/id_rsa.pub.
e. Nos conectaremos a la máquina con centos para habilitar temporalmente el login con contraseña. Para ello editamos con sudo el siguiente fichero **/etc/ssh/sshd_config**, cambiando **PasswordAuthentication** por **yes**. Una vez cambiado reiniciamos el servicio sshd: **sudo systemctl restart sshd**.
f. Configuraremos una contraseña para el usuario vagrant con: **sudo passwd vagrant**.
g. Añadir la Ip de la centos al fichero /etc/hosts
h. Volveremos a la máquina ubuntu y usaremos el comando **ssh-copy-id centos** para copiar la clave pública que acabamos de crear. Deberemos introducir la contraseña del usuario vagrant que acabamos de crear.
i. Después de copiar la clave pública, desharemos los cambios que hemos hecho en el apartado e y reiniciamos de nuevo el servicio sshd.
j. Ahora probaremos conectarnos desde la máquina ubuntu por ssh a la máquina centos, y no nos debería pedir contraseña.

13. Instalar servidor web apache y nginx

a. Seguiremos la siguiente guía para instalar y configurar Apache: https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-16-04
b. Seguiremos la siguiente guía para instalar y configurar nginx: https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04-quickstart

14. Sniffer de paquetes con tcpdump

a. Conectarse a la máquina con ubuntu y pasarse a root
b. Instalar el paquete tcpdump con **apt-get install tcpdump**.
c. Capturar tráfico con el comando **tcpdump -i enp0s3 -nn -s0 -v port 80**

d. Abrimos otra terminal contra la máquina ubuntu y ejecutamos **curl -s www.google.com > /dev/null**.
e. ¿que aparece en la salida del tcpdump?
f. Vamos a capturar información sobre el User-Agent y el Host, para ello ejecutamos **tcpdump -i enp0s3 -nn -s1500 -A -l | egrep -i 'User-Agent:|Host:'** y volvemos a ejecutar el mismo curl desde otra terminal.
g. ¿Obtenemos el resultado esperado?
h. Vamos a capturar otro tipo de tráfico, como podría ser ICMP (pings). Para ello lanzamos el tcpdump de la siguiente manera: **tcpdump -i enp0s3 -n icmp** y desde otro terminal ejecutamos **ping 8.8.8.8.**
i. ¿Podemos ver el tráfico que se ha generado?
j. Vamos a almacenar la salida del tcpdump en un fichero para luego analizarlo con Wireshark gráficamente. Para ello lo ejecutamos con **tcpdump -i enp0s3 -n icmp -w /vagrant/ping.pcap**
k. Volvemos a lanzar el ping y en /vagrant/ping.pcap tendremos las trazas que podremos abrir con wireshark.

15. Uso del comando ss

a. El comando ss nos devuelve información sobre los sockets abiertos en el sistema. Nos conectaremos a alguna de las dos máquinas que hemos usado en los ejercicios anteriores. Ejecutando ss obtenemos información sobre todos los sockets, TCP, UDP y UNIX.
b. Podemos usar los parámetros -t, -u o -x para filtrar por TCP, UDP o Unix respectivamente.
c. Con **ss dst 10.0.2.2** podemos obtener información sobre los sockets establecidos contra un destino específico.
d. Si en nuestro servidor tenemos algún servicio escuchando en algún puerto lo veremos con **ss -l**.
e. Como vemos, en los puertos se especifica el nombre del servicio que corresponde, con **ss -ln** no hace esta substitución y vemos el valor numérico del puerto.
f. También podemos ver información del proceso que ha definido el socket añadimos la opción -p: **ss -tlnp**

16. Verificar conectividad entre dos puertos con nc

a. Levantar las dos máquinas y conectarse mediante ssh a cada una de ellas
b. En una de ellas levantar un socket escuchando en el puerto 4444 con el comando **nc -l 4444 &**
c. Verificar con **ss** si el estado del puerto 4444
d. Desde otra máquina verificar la conectividad con ese puerto con el comando **nc <IP> 4444**. Escribir algo en esa terminal y verificar que la máquina que está escuchando recibe los datos
e. Verificar con **ss** el estado del puerto

f. Cerrar la conexión desde el cliente y cerrará también la del servidor.
g. ¿En qué estado han quedado los puertos?

17. Montar un NFS entre dos máquinas

a. On server1 (centos), use **yum install nfs-utils policycoreutils-python -y** to install the required utilities.
b. Open the file /etc/exports with an editor and add the following line:
   /srv/nfsexport *(rw)
c. Type **mkdir /srv/nfsexport** to create the shared directory.
d. Change the ownership of the new folder with: **chown -R nfsnobody:nfsnobody /srv**
e. Set the NFS context on the share:
   semanage fcontext -a -t nfs_t "/srv/nfsexport(/.*)?".
   Type **restorecon -Rv /srv/nfsexport** to apply the setting to the file system.
f. Launch and enable the NFS server by using **systemctl enable nfs-server --now;**
g. From server2 (ubuntu), install the following package **nfs-utils** with the apt command.
h. Ensure that from server2 you can reach server1 by IP.
i. Type **showmount -e server1-ip**.
j. **11.** On server2, type **mkdir /mnt/nfs;** and mount the shared filesystem with **mount server1-ip:/srv/nfsexport /mnt/nfs**.
k. Type **mount** to verify that the NFS share has mounted correctly.
l. Still on server2, open the file /etc/fstab and add the following line to make the mount persistent:
   server1-ip:/srv/nfsexport /mnt/nfs nfs _netdev 0 0
m. **14.** Type **systemctl status remote-fs.target** and verify that this systemd unit is enabled.
n. **15.** Restart server2 with shutdown -r now and verify that the mount is activated automatically (ensure that you have an static IP configuration for the internal network).

18. Curl

a. Algunos ejemplos de curl:
   curl http://wttr.in
   curl v2.wttr.in
   curl wttr.in/Moon
   curl v2.wttr.in/Moon
   curl qrenco.de
   curl cheat\.sh/curl
   curl cheat.sh/curl
   curl cheat.sh/bash
   curl cheat.sh/latencies

```
curl cheat.sh/chmod
curl cheat.sh/mkdir
curl cheat.sh/systemctl
curl cheat.sh/journalctl
```