Tecnología NetPlan para la configuración de red en sistemas Linux

Fundamentos de NetworkManager

Remover NetPlan y volver a ifupdown

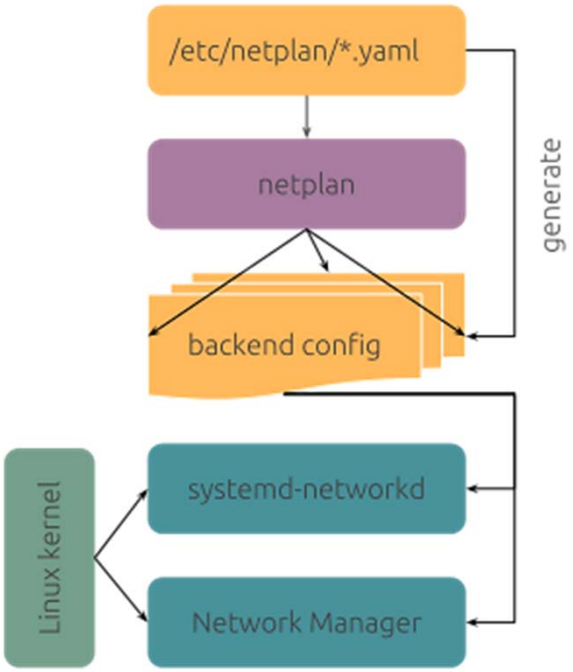
https://netplan.io/

04-TCP-IP-Básico_Ext-NetPlan.pptx
Alfredo Abad
UA: 26-feb-2019

¿Qué es NetPlan?

- Es una abstracción de la configuración de la red para sistemas Linux
- NetPlan lee la información de configuración proporcionada por el usuario de los ficheros *.yaml (escritos en formato yaml) de las siguientes ubicaciones y por este orden:
  - /lib/netplan/*.yaml
  - /etc/netplan/*.yaml
  - /run/netplan/*.yaml
- Durante el inicio de sistema, Netplan genera las configuraciones de red necesarias en /run para dos posibles sistemas gestores de red (también llamados “renderers”):
  - NetworkManager
  - Systemd-networkd
- La arquitectura de Netplan se especifica en la diapo siguiente

Arquitectura de NetPlan



3
Alfredo Abad

Configuración básica

```
network:
  version: 2
  renderer: NetworkManager
```

- El fichero yaml contiene lo que se especifica arriba
- Significa
  - Todas las tarjetas de red Ethernet tomarán su configuración del DHCP en cuanto detecten portadora
  - El sistema gestor para todas las tarjetas será NetworkManager
- Nota:
  - Si hubiéramos elegido el renderer “networkd”, habría que haber definido cada tarjeta específicamente (para NetworkManager no es necesario, pero para networkd sí)

4
Alfredo Abad

Los dos comandos básicas en NetPlan

- Para regular su comportamiento a partir de los ficheros de configuración, NetPlan utiliza dos subcomandos

- **netplan generate**
  - Usa /etc/netplan/*.yaml para generar la configuración requerida para los renderers configurados en esos ficheros yaml
- **netplan apply**
  - Aplica la configuración generada para los renderers, restableciéndolos si fuera necesario para que el sistema operativo tome en tiempo real las configuraciones programadas
- Se puede probar la validez de la configuración antes de hacer el apply con **netplan try**

- NetPlan no genera configuraciones persistentes, ya que se generan cada vez que se inicia el sistema
- Si un dispositivo aparece configurado en un yaml (por ejemplo en /lib/netplan) y luego se encuentra otro (en /etc/netplan o /run/netplan), la información se añade de uno a otro (amend), como si todas las claves de configuración estuvieran en un único fichero

5
Alfredo Abad

Ejemplo de netplan: Configurar IP estática con networkd

Como ya hemos comentado en otros posts, **netplan** utiliza archivos de configuración en formato **yaml** almacenados en el directorio **/etc/netplan**. Y si no hay ningún archivo de configuración en **/etc/netplan**, podemos generarlo automáticamente con tan sólo ejecutar:

```
# netplan generate
```

Una vez generado, podemos editarlo para ajustar la configuración.

En máquinas conectadas vía ethernet, en las que no necesitamos que el usuario pueda gestionar la configuración de red, podemos usar el renderer **networkd** en lugar de **NetworkManager**.

Para entender mejor cómo configurar una IP estática, vamos a ver un ejemplo:

```
# cat /etc/netplan/01-network-manager-all.yaml
```

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s1:
      dhcp4: no
      dhcp6: no
      adresses: [172.19.144.16/23]
      gateway4: 172.19.144.1
      nameservers:
         addresses: [172.19.144.2,172.19.144.3,208.67.222.222]
```

6
Alfredo Abad

```
# cat /etc/netplan/01-network-manager-all.yaml
```

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s1:
      dhcp4: no
      dhcp6: no
      adresses: [172.19.144.16/23]
      gateway4: 172.19.144.1
      nameservers:
         addresses: [172.19.144.2,172.19.144.3,208.67.222.222]
```

Como podéis observar, en este ejemplo, estamos configurando:

- Una tarjeta de red cuyo nombre es **enp0s1** y estamos especificando que no debe obtener una dirección IP mediante dhcp (dhcp4: no, dhcp6:no).
- Una dirección ip estática junto con la máscara de red (172.19.144.16/23).
- Un gateway para IPv4 (172.19.144.1)
- Diferentes servidores DNS (addresses: [172.19.144.2,172.19.144.3,208.67.222.222])

No olvidéis que es necesario respetar los sangrados en el fichero de configuración **.yaml**.

Una vez realizados los cambios en el fichero de configuración, comprobamos que no hay errores en la configuración:

```
# netplan try
```

Y si no hay errrores, aplicamos la configuración:

```
# netplan apply
```

7
Alfredo Abad

Otro ejemplo, expresado por fases

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
```

```
network:
  version: 2
  renderer: networkd
```

```
ethernets:
eno1:
  addresses:
    - 192.168.1.142/24
  gateway4: 192.168.1.1
  nameservers:
    addresses:
        - 192.168.1.1
        - 8.8.8.8
        - 8.8.4.4
  dhcp4: no
  dhcp6: no
```

8
Alfredo Abad

Ejemplo (continuación): Wifis y bridges

```
wifis:
wlp53s0:
  addresses:
    - 192.168.1.233/24
  gateway4: 192.168.1.1
  nameservers:
    addresses:
        - 192.168.1.1
        - 8.8.8.8
        - 8.8.4.4
  dhcp4: no
  dhcp6: no
  access-points:
    MyPoint:
      password: STRONG_PASSWORD_PLS
```

```
bridges:
br0:
  interfaces:
      - eno1
  dhcp4: yes
```

9
Alfredo Abad

Ejemplo (todo junto)

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eno1:
      addresses:
                - 192.168.1.142/24
      gateway4: 192.168.1.1
      nameservers:
              addresses:
                  - 192.168.1.1
                  - 8.8.8.8
                  - 8.8.4.4
      dhcp4: no
      dhcp6: no
  wifis:
    wlp58s0:
      addresses:
                - 192.168.1.233/24
      gateway4: 192.168.1.1
      nameservers:
              addresses:
                  - 192.168.1.1
                  - 8.8.8.8
                  - 8.8.4.4
      dhcp4: no
      dhcp6: no
      access-points:
        Your_access_point:
          password: STRONG_PASSWORD_PLS
  bridges:
    br0:
      interfaces:
          - eno1
      dhcp4: yes
```

10
Alfredo Abad

Testeo y aplicación

- Testeo de la configuración

```
1 sudo netplan --debug generate
```

  - Otra forma de test (sin verbosidad): **sudo netplan try**

- Aplicación de la configuración

```
1 sudo netplan apply
```

11
Alfredo Abad

Estructura el fichero yaml para NetPlan

12
Alfredo Abad

La cabecera de un yaml de NetPlan

- MUY IMPORTANTE: siempre hay que respetar las identaciones en el fichero
- El primer nodo de un yaml de NetPlan es la palabra reservada (mapping) "**network:"**
- Sigue la especificación de la versión, que será la 2 (algunos sistemas de especificación de red usan la versión 1: MaaS, curtin, etc.)
  - **version: 2**
- Seguidamente se especifican, agrupados por sus tipos (Configuration IDs), las definiciones de dispositivos (que deben ser admitidos por el backend que vaya a procesarlos). Por ejemplo:
  - ethernets:
  - wifis:
  - bridges:

13
Alfredo Abad

Propiedades para los tipos de dispositivos físicos

- match (mapping)
  - Permite especificar uno o varios dispositivos por un patrón
  - Permite múltiples patrones, todos las cuales deben cumplirse en el/los dispositivo/s seleccionado/s
- Algunos de estos patrones son:
  - name (scalar) Para el nombre del dispositivo
  - macaddress (scalar) Para la dirección MAC del dispositivo
  - driver (scalar) Para elegir dispositivos por el nombre de su driver en kernel
  - set-name (scalar)
    - Al hacer coincidir propiedades únicas como ruta o MAC, o con supuestos adicionales como "solo habrá un dispositivo wifi", las reglas de coincidencia pueden escribirse de modo que solo coincidan con un dispositivo. Entonces, esta propiedad se puede usar para darle a ese dispositivo un nombre más específico / deseable / más agradable que el predeterminado de los nombres de usuario de udev
  - wakeonlan (bool) Habilita wake on lan (por defecto está en OFF)

14
Alfredo Abad

Ejemplos

- all cards on second PCI bus:

```
match:
  name: enp2*
```

- fixed MAC address:

```
match:
  macaddress: 11:22:33:AA:BB:FF
```

- first card of driver `ixgbe`:

```
match:
  driver: ixgbe
  name: en*s0
```

15
Alfredo Abad

Propiedades comunes a todos los dispositivos (no necesariamente físicos; solo se especifican los habituales)

- renderer (scalar) Actualmente soporta NetworkManager y networkd
- dhcp4 (bool) Habilita DHCP para IPv4 (por defecto, OFF)
- dhcp6 (bool) Ídem para IPv6
- accept-ra (bool) Acepta Router Advertisement en IPv6 (ON)
- addresses (sequence of scalars)
  - Agrega direcciones estáticas a la interfaz además de las recibidas a través de DHCP o RA. Cada entrada de secuencia está en notación CIDR,
  - Ejemplo: **addresses: [192.168.14.2/24, "2001:1::1/64"]**
- gateway4, gateway6 (scalar)
  - Habilita la puerta por defecto en IPv4 e IPv6
  - Ejemplo: **gateway4: 172.16.0.1 Example for IPv6: gateway6: "2001:4::1“**

16
Alfredo Abad

Propiedades comunes a todos los dispositivos (II)

- nameservers (mapping) Define los resolvedores de nombres
  - Tiene dos subcampos: search (nombres de dominio de búsqueda) y addresses (direcciones IP de los resolvedores)

```
ethernets:
  id0:
    [...]
    nameservers:
      search: [lab, home]
      addresses: [8.8.8.8, "FEDC::1"]
```

17
Alfredo Abad

Propiedades comunes a todos los dispositivos (III)

- optional (bool) El dispositivo es o no necesario para el arranque
- routes (mapping) Configura una ruta estática para el dispositivo
- routing-policy (mapp.) Configura una “policy routing”
- macaddress (scalar) Establece MAC (formato: “XX:XX:XX:XX:XX:XX”)

```
ethernets:
  id0:
    [...]
    macaddress: 52:54:00:6b:3c:59
```

18
Alfredo Abad

Ejemplo

```
ethernets:
  eth7:
    # this is plugged into a test network that is often
    # down - don't wait for it to come up during boot.
    dhcp4: true
    optional: true
```

19
Alfredo Abad

Especificación del ROUTING con routes

- routes (mapping) Requiere al menos la opción via, el resto son opcionales
- Opciones de la especificación “routes”
  - from (scalar) IP origen del tráfico
  - to (scalar) IP destino del tráfico
  - via (scalar) IP de ruta
  - on-link (bool) True: especifica que la ruta está conectada directamente al interfaz
  - metric (scalar) Prioridad de la ruta (debe ser un entero positivo)
  - type (scalar) Tipo de ruta: unicast (def.), unreachable, blackhole o prohibited
  - scope (scalar) Ámbito de la ruta: global, link o host
  - table (scalar) Número de tabla de ruta (tomado de /etc/iproute2/rt_tables)

20
Alfredo Abad

Especificación del ROUTING con routing-policy

- routing-policy (mapping)
- Opciones de la especificación “routing-policy”
  - from (scalar) IP de origen para aplicar la política de rutas especificada
  - to (scalar) IP de destino
  - table (scalar) Número de tabla de ruta (tomado de /etc/iproute2/rt_tables)
  - priority (scalar) Prioridad de regla de la política (números altos, menor prioridad)
  - fwmark (scalar) Número de marca del tráfico marcado por iptables
  - type-of-service (scalar) Tipo de servicio requerido para usar la política

21
Alfredo Abad

Propiedades para dispositivos Ethernet (ethernets:) y WiFi (wifis:)

- Los dispositivos ethernets no soportan ninguna especificación ulterior
- systemd-networkd no soportan nativamente las WIFIs, de modo que en este caso es necesaria la instalación de un wpasupplicant
- NetworkManager sí soporta nativamente las wifis
- Especificaciones wifis:
  - access-points (mapping) APs preconfigurados para NetworkManager
  - password (scalar) Passphrase para autenticación WPA2 (open, en vacío)
  - mode (scalar)
    - Los posibles modos de puntos de acceso son **infrastructure** (por defecto), **ap** (crear un punto de acceso al que se pueden conectar otros dispositivos) y **adhoc** (redes punto a punto sin un punto de acceso central). ap solo es compatible con NetworkManager.

22
Alfredo Abad

Propiedades para dispositivos de tipo bridges: (I)

- interfaces (sequence of scalars) Dispositivos que componen el bridge

```
ethernets:
  switchports:
    match: {name: "enp2*"}
[...]
bridges:
  br0:
    interfaces: [switchports]
```

23
Alfredo Abad

Propiedades para dispositivos de tipo bridges: (II)

- parameters (mapping) Abre parámetros para bridges (los siguientes)
- ageing-time (scalar) Tiempo de custodia de la MAC address
- priority (scalar) Prioridad del bridge para la elección del root
- port-priority (scalar) Prioridad del puerto (de 0 a 63)
- forward-delay (scalar) Tiempo de escucha en aprendizajes
- hello-time (scalar) Intervalo entre dos hellos de nodo root a otros
- max-age (scalar) Edad máxima de un hello packet
- path-cost (scalar) Coste del camino del bridge
- stp (bool) True (defecto), habilita el protocolo STP

- Nota: los parámetros de tiempo se especifican en milisegundos para NetworkManager y en segundos para systemd

24
Alfredo Abad

Propiedades para dispositivos de tipo bonds: (I)

- interfaces (sequence of scalars)

```
ethernets:
  switchports:
    match: {name: "enp2*"}
[...]
bonds:
  bond0:
    interfaces: [switchports]
```

25
Alfredo Abad

Propiedades para dispositivos de tipo bonds: (II)

- parameters (mapping) mode (scalar)
- lacp-rate (scalar)
- mii-monitor-interval (scalar)
- min-links (scalar)
- transmit-hash-policy (scalar)
- ad-select (scalar)
- all-slaves-active (bool)
- arp-interval (scalar)
- arp-ip-targets (secuence of scalars)
- arp-validate (scalar)
- arp-all-targets (scalar)
- up-delay (scalar)
- down-delay (scalar)
- fail-over-mac-policy (scalar)
- gratuitious-arp (scalar)
- packets-per-slave (scalar)
- primary-reselect-policy (scalar)
- resend-igmp (scalar)
- learn-packet-interval (scalar)
- primary (scalar)

26
Alfredo Abad

Propiedades para dispositivos vlans:

- id (scalar) VLAN ID, un entero entre 0 y 4094
- link (scalar) NetPlan ID (dispositivo) sobre el que se crea la VLAN

```
ethernets:
  eno1: {...}
vlans:
  en-intra:
    id: 1
    link: eno1
    dhcp4: yes
  en-vpn:
    id: 2
    link: eno1
    address: ...
```

27
Alfredo Abad

EJEMPLOS

28
Alfredo Abad

Configure un dispositivo ethernet con networkd, identificado por su nombre, y habilite DHCP

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp3s0:
      dhcp4: true
```

29
Alfredo Abad

IP estática, con resolvedores y puerta por defecto

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp3s0:
      addresses:
        - 10.10.10.2/24
      gateway4: 10.10.10.1
      nameservers:
          search: [mydomain, otherdomain]
          addresses: [10.10.10.1, 1.1.1.1]
```

30
Alfredo Abad

Ejemplo con WiFi

```
network:
  version: 2
  renderer: networkd
  wifis:
    wlp2s0b1:
      dhcp4: no
      dhcp6: no
      addresses: [192.168.0.21/24]
      gateway4: 192.168.0.1
      nameservers:
        addresses: [192.168.0.1, 8.8.8.8]
      access-points:
        "network_ssid_name":
          password: "**********"
```

31
Alfredo Abad

Múltiples IP en una interfaz

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp3s0:
      addresses:
        - 10.100.1.38/24
        - 10.100.1.39/24
      gateway4: 10.100.1.1
```

32
Alfredo Abad

Múltiples IP con múltiples gateways

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp3s0:
      addresses:
        - 9.0.0.9/24
        - 10.0.0.10/24
        - 11.0.0.11/24
      #gateway4:    # unset, since we configure routes below
      routes:
        - to: 0.0.0.0/0
          via: 9.0.0.1
          metric: 100
        - to: 0.0.0.0/0
          via: 10.0.0.1
          metric: 100
        - to: 0.0.0.0/0
          via: 11.0.0.1
          metric: 100
```

33
Alfredo Abad

Dos interfaces de red con dhcp

```
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: yes
    eth1:
      dhcp4: yes
```

34
Alfredo Abad

Múltiples IP y gateways

```
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
 version: 2
 renderer: networkd
 ethernets:
  eth0:
   dhcp4: no
   dhcp6: no
   addresses:
    - 81.169.XXX.YY/32
    - 81.169.YYY.XXX/32
    - "2a01:238:42dd:dddd:aaaa:cccc:yyyy:xxxx/128"
   gateway4: 81.169.1YY.XX
   gateway6: fe80::1
   nameservers:
    addresses: [81.169.163.106, 85.214.7.22, 81.169.148.34]
   routes:
    - to: 0.0.0.0/0
      via: 81.169.1YY.XX
      on-link: true
```

35
Alfredo Abad

Una interfaz configurada como puente

```
auto lo
iface lo inet loopback

auto enp10s0
iface enp10s0 inet manual

auto br0
iface br0 inet dhcp
    bridge_ports enp10s0
    bridge_stp off
    bridge_fd 0
    bridge_maxwait 0
```

36
Alfredo Abad

NetworkManager (izqda.) y Bonding (dcha.)

```
network:
  version: 2
  renderer: NetworkManager
```

```
network:
  version: 2
  renderer: networkd
  bonds:
    bond0:
      dhcp4: yes
      interfaces:
        - enp3s0
        - enp4s0
      parameters:
        mode: active-backup
        primary: enp3s0
```

37
Alfredo Abad

Bonding actuando como router (con interfaces opcionales)

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp1s0:
      dhcp4: no
    enp2s0:
      dhcp4: no
    enp3s0:
      dhcp4: no
      optional: true
    enp4s0:
      dhcp4: no
      optional: true
    enp5s0:
      dhcp4: no
      optional: true
    enp6s0:
      dhcp4: no
      optional: true
  bonds:
    bond-lan:
      interfaces: [enp2s0, enp3s0]
      addresses: [192.168.93.2/24]
```

```
bonds:
  bond-lan:
    interfaces: [enp2s0, enp3s0]
    addresses: [192.168.93.2/24]
    parameters:
      mode: 802.3ad
      mii-monitor-interval: 1
  bond-wan:
    interfaces: [enp1s0, enp4s0]
    addresses: [192.168.1.252/24]
    gateway4: 192.168.1.1
    nameservers:
      search: [local]
      addresses: [8.8.8.8, 8.8.4.4]
    parameters:
      mode: active-backup
      mii-monitor-interval: 1
      gratuitious-arp: 5
  bond-conntrack:
    interfaces: [enp5s0, enp6s0]
    addresses: [192.168.254.2/24]
    parameters:
      mode: balance-rr
      mii-monitor-interval: 1
```

38
Alfredo Abad

Bridging

```
network:
  version: 2
  renderer: networkd
  bridges:
    br0:
      dhcp4: yes
      interfaces:
        - enp3s0
```

39
Alfredo Abad

Ejemplo de bridge

Para poder usar bridges, lo primero que haremos será instalar el paquete **bridge-utils**:

```
# apt update && apt -y install bridge-utils
```

Supongamos que tenemos una máquina con la siguiente configuración de red:

```
# cat /etc/netplan/01-systemd-networkd.yaml
```

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eno1:
      dhcp4: yes
      dhcp6: no
```

Para configurar el bridge (br0), editaremos el fichero /etc/netplan/01-systemd-networkd.yaml y lo dejaremos así:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eno1:
      dhcp4: no
      dhcp6: no
  bridges:
    br0:
      interfaces: [eno1]
      dhcp4: yes
      dhcp6: no
```

Si comparáis el contenido actual con el anterior, veréis que hemos deshabilitado dhcp4 en la interfaz eno01 y lo hemos activado en el bridge (br0).

También podéis observar que en las línea interfaces: [eno1] estamos definiendo que el bridge br0 está creado con la interfaz eno1. Si quisiéramos implementar el bridge con varias interfaces, los nombres de las diferentes interfaces irían dentro de los corchetes, separarados por comas. Por ejemplo: [eno1, enp3s0].

Una vez modificado el fichero de configuración ejecutamos:

```
# netplan generate
```

Y

```
# netplan apply
```

40
Alfredo Abad

Bridging complejo con vlan TAG y UNTAG sobre lbvirtd

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s25:
      dhcp4: true
  bridges:
    br0:
      addresses: [ 10.3.99.25/24 ]
      interfaces: [ vlan15 ]
  vlans:
    vlan15:
      accept-ra: no
      id: 15
      link: enp0s25
```

Then libvirtd would be configured to use this bridge by adding the following content to a new XML file under `/etc/libvirtd/qemu/networks/`. The name of the bridge in the tag as well as in need to match the name of the bridge device configured using netplan:

```
<network>
  <name>br0</name>
  <bridge name='br0'/>
  <forward mode="bridge"/>
</network>
```

VLAN

```
network:
  version: 2
  renderer: networkd
  ethernets:
    mainif:
      match:
        macaddress: "de:ad:be:ef:ca:fe"
      set-name: mainif
      addresses: [ "10.3.0.5/23" ]
      gateway4: 10.3.0.1
      nameservers:
        addresses: [ "8.8.8.8", "8.8.4.4" ]
        search: [ example.com ]
  vlans:
    vlan15:
      id: 15
      link: mainif
      addresses: [ "10.3.99.5/24" ]
    vlan10:
      id: 10
      link: mainif
      addresses: [ "10.3.98.5/24" ]
      nameservers:
        addresses: [ "127.0.0.1" ]
        search: [ domain1.example.com, domain2.example.com ]
```

42
Alfredo Abad

Directly connected Gateway

This allows setting up a default route, or any route, using the "on-link" keyword where the gateway is an IP address that is directly connected to the network even if the address does not match the subnet configured on the interface.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    addresses: [ "10.10.10.1/24" ]
    routes:
      - to: 0.0.0.0/0
        via: 9.9.9.9
        on-link: true
```

43
Alfredo Abad

Source routing

Route tables can be added to particular interfaces to allow routing between two networks:

In the example below, ens3 is on the 192.168.3.0/24 network and ens5 is on the 192.168.5.0/24 network. This enables clients on either network to connect to the other and allow the response to come from the correct interface.

Furthermore, the default route is still assigned to ens5 allowing any other traffic to go through it.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    ens3:
      addresses:
       - 192.168.3.30/24
      dhcp4: no
      routes:
       - to: 192.168.3.0/24
         via: 192.168.3.1
         table: 101
      routing-policy:
       - from: 192.168.3.0/24
         table: 101
    ens5:
      addresses:
       - 192.168.5.24/24
      dhcp4: no
      gateway4: 192.168.5.1
      routes:
       - to: 192.168.5.0/24
         via: 192.168.5.1
         table: 102
      routing-policy:
        - from: 192.168.5.0/24
          table: 102
```

44
Alfredo Abad

Loopback

Networkd does not allow creating new loopback devices, but a user can add new addresses to the standard loopback interface, lo, in order to have it considered a valid address on the machine as well as for custom routing:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    lo:
      match:
        name: lo
      addresses: [ 7.7.7.7/32 ]
```

45
Alfredo Abad

Windows DHCP Server

For networks where DHCP is provided by a Windows Server using the dhcp-identifier key allows for interoperability:

```
network:
  version: 2
  ethernets:
    enp3s0:
      dhcp4: yes
      dhcp-identifier: mac
```

46
Alfredo Abad

Un ejemplo, complejo y más completo

```
network:
  version: 2
  # if specified, can only realistically have that value, as networkd cannot
  # render wifi/3G.
  renderer: NetworkManager
  ethernets:
    # opaque ID for physical interfaces, only referred to by other stanzas
    id0:
      match:
        macaddress: 00:11:22:33:44:55
      wakeonlan: true
      dhcp4: true
      addresses:
        - 192.168.14.2/24
        - 192.168.14.3/24
        - "2001:1::1/64"
      gateway4: 192.168.14.1
      gateway6: "2001:1::2"
      nameservers:
        search: [foo.local, bar.local]
        addresses: [8.8.8.8]
```

```
    gateway4: 192.168.14.1
    gateway6: "2001:1::2"
    nameservers:
      search: [foo.local, bar.local]
      addresses: [8.8.8.8]
    routes:
      - to: 0.0.0.0/0
        via: 11.0.0.1
        table: 70
        on-link: true
        metric: 3
    routing-policy:
      - to: 10.0.0.0/8
        from: 192.168.14.2/24
        table: 70
        priority: 100
      - to: 20.0.0.0/8
        from: 192.168.14.3/24
        table: 70
        priority: 50
  lom:
    match:
```

47 Alfredo Abad

```
      - to: 20.0.0.0/8
        from: 192.168.14.3/24
        table: 70
        priority: 50
  lom:
    match:
      driver: ixgbe
    # you are responsible for setting tight enough match rules
    # that only match one device if you use set-name
    set-name: lom1
    dhcp6: true
  switchports:
    # all cards on second PCI bus unconfigured by
    # themselves, will be added to br0 below
    # note: globbing is not supported by NetworkManager
    match:
      name: enp2*
    mtu: 1280
ifis:
  all-wlans:
    # useful on a system where you know there is
    # only ever going to be one device
    match: {}
    access-points:
```

```
wifis:
  all-wlans:
    # useful on a system where you know there is
    # only ever going to be one device
    match: {}
    access-points:
      "Joe's home":
        # mode defaults to "infrastructure" (client)
        password: "s3kr1t"
  # this creates an AP on wlp1s0 using hostapd
  # no match rules, thus the ID is the interface name
  wlp1s0:
    access-points:
      "guest":
        mode: ap
        # no WPA config implies default of open
bridges:
  # the key name is the name for virtual (created) interfaces
  # no match: and set-name: allowed
  br0:
    # IDs of the components; switchports expands into multiple interfaces
    interfaces: [wlp1s0, switchports]
    dhcp4: true
```

48 Alfredo Abad

Gestión de NetworkManager

Instalación de NetworkManager y sus plugins

- Instalación del core
  - sudo apt-get install network-manager
- Instalación del applet gráfico de GNOME para NetworkManager
  - sudo apt-get install network-manager-gnome
- Instalación de algunos plugins para vpn (core)
    - sudo apt-get install network-manager-pptp (viene instalado por defecto)
  - sudo apt-get install network-manager-openvpn
  - sudo apt-get install network-manager-vpnc
  - sudo apt-get install network-manager-openconnect
- Instalación de algunos plugins para vpn (paquete para GNOME)
  - sudo apt-get install network-manager-openvpn
  - sudo apt-get install network-manager-vpnc
  - sudo apt-get install network-manager-openconnect

Inicio y parada de NetworkManager

- Con Upstart
  - **sudo start network-manager** (el servicio se habilita en inicio, por defecto)
  - **sudo stop network-manager**
  - Crear un fichero de override para Upstart y que se deshabilite el servicio en inicio
    - **echo "manual" | sudo tee /etc/init/network-manager.override**
- Con Systemd
  - **sudo systemctl start NetworkManager.service**
  - **sudo systemctl stop NetworkManager.service**
- Habilitar el servicio para que arranque en inicio con Systemd
  - **sudo systemctl enable NetworkManager.service**
- Deshabilitar el servicio en arranque con Systemd
  - **sudo systemctl disable NetworkManager.service**

51
Alfredo Abad

Localización de ficheros de configuración

- Todos los ficheros de configuración se almacenan en los siguientes dos directorios:
  - /etc/NetworkManager
  - /etc/NetworkManager/system-connections
- NetworkManager manejará todas las conexiones del sistema
  - Habría que comentar todas las líneas en /etc/network/interfaces, excepto las líneas de la interfaz lo

52
Alfredo Abad

Remover NetPlan para volver al sistema antiguo con ifupdown

https://askubuntu.com/questions/977243/ubuntu-17-10-disable-netplan

53
Alfredo Abad

ifupdown: volveremos a configurar con /etc/network/interfaces y a utilizar sus herramientas asociadas

- Chequear los nombres de interfaces de red y sus direcciones (si es de interés), con **ip l** e **ip a**
- Instalar ifupdown con **sudo apt -y install ifupdown**
  - Quizá también interese instalar el paquete **net-tools**
- Purgar netplan con **sudo apt -y purge netplan.io**
- Configurar **/etc/network/interfaces** y/o **/etc/network/interfaces.d** de acuerdo con las necesidades
- Reiniciar el servicio de red con **sudo systemctl restart networking**; **systemctl status networking** o con **sudo /etc/init.d/networking restart**; **/etc/init.d/networking status**
- Opcionalmente, purgar manualmente los remanentes de los ficheros de configuración de netplan con **sudo rm -vfr /usr/share/netplan /etc/netplan**

54
Alfredo Abad

Más completo

https://askubuntu.com/questions/1031709/ubuntu-18-04-switch-back-to-etc-network-interfaces

I. Reinstall the **ifupdown** package:

```
# apt-get update
# apt-get install ifupdown
```

II. Configure your **/etc/network/interfaces** file with configuration stanzas such as:

```
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

allow-hotplug enp0s3
auto enp0s3
iface enp0s3 inet static
  address 192.168.1.133
  netmask 255.255.255.0
  broadcast 192.168.1.255
  gateway 192.168.1.1
  # Only relevant if you make use of RESOLVCONF(8)
  # or similar...
  dns-nameservers 1.1.1.1 1.0.0.1
```

III. Make the configuration effective (no reboot needed):

```
# ifdown --force enp0s3 lo && ifup -a
# systemctl unmak networking
# systemctl enable networking
# systemctl restart networking
```

IV. Disable and remove the unwanted services:

```
# systemctl stop systemd-networkd.socket systemd-networkd \
networkd-dispatcher systemd-networkd-wait-online
# systemctl disable systemd-networkd.socket systemd-networkd \
networkd-dispatcher systemd-networkd-wait-online
# systemctl mask systemd-networkd.socket systemd-networkd \
networkd-dispatcher systemd-networkd-wait-online
# apt-get --assume-yes purge nplan netplan.io
```

55
Alfredo Abad