[Redes] Configuración en Linux usando Netplan

- Como configurar la red con Netplan en Ubuntu
  - Fichero de configuración en `/etc/netplan/algo.yaml`. El archivo de configuración o los archivos los podemos listar con

    ```
    $ ls /etc/netplan/
    ```

    en mi caso `algo.yaml = 01-network-manager-all.yaml`.
  - Con

    ```
    $ ip a
    ```

    obtendremos información de nuestras interfaces de red. En mi caso aparecen dos con los nombres de `lo` y `ens3`.
  - Editar el fichero de configuración, por ejemplo con

    ```
    nano /etc/netplan/01-network-manager-all.yaml
    ```

    y poner la configuración deseada. Para ello se puede consultar Netplan configuration examples
- Obtener toda nuestra configuración de red con comandos
- Cómo usar el comando IP (el sustituto de ifconfig en Debian 9)

Configuración

Ejemplos de configuración de red con Netplan. La información que aquí se cita proviene principalmente de Netplan configuration examples.

Para configurar netplan, guarde los archivos de configuración `/etc/netplan/`con una extensión `.yaml` (por ejemplo `/etc/netplan/config.yaml`), luego ejecute `sudo netplan apply`. Este comando analiza y aplica la configuración al sistema. La configuración escrita en el disco en `/etc/netplan/` persistirá entre reinicios.

Usando DHCP y direccionamiento estático

IP dinámica. Para permitir que la interfaz llamada `enp3s0` obtenga una dirección a través de DHCP, cree un archivo YAML con lo siguiente:

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

IP estática. En su lugar, para establecer una dirección IP estática, use la clave de direcciones, que toma una lista de (IPv4 o IPv6), direcciones junto con la longitud del prefijo de subred (por ejemplo, / 24). También se puede proporcionar información sobre la puerta de enlace y el DNS:

```
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]
```

Conexión de múltiples interfaces con DHCP

Muchos sistemas ahora incluyen más de una interfaz de red. Los servidores normalmente necesitarán conectarse a múltiples redes y pueden requerir que el tráfico a Internet pase a través de una interfaz específica a pesar de que todos ellos proporcionan una puerta de enlace válida.

Se puede lograr el enrutamiento exacto deseado a través de DHCP especificando una métrica para las rutas recuperadas a través de DHCP, lo que garantizará que algunas rutas se prefieran sobre otras. En este ejemplo, se prefiere `enred` sobre `engreen`, ya que tiene una métrica de ruta más baja:

```
network:
    version: 2
    ethernets:
        enred:
            dhcp4: yes
            dhcp4-overrides:
                route-metric: 100
        engreen:
            dhcp4: yes
            dhcp4-overrides:
                route-metric: 200
```

Conexión a una red inalámbrica abierta

Netplan admite fácilmente la conexión a una red inalámbrica abierta (una que no está protegida por una contraseña), solo requiere que el punto de acceso esté definido:

```
network:
    version: 2
    wifis:
        wl0:
            access-points:
                opennetwork: {}
            dhcp4: yes
```

Conexión a una red inalámbrica personal WPA

Los dispositivos inalámbricos utilizan la clave 'wifis' y comparten las mismas opciones de configuración con los dispositivos ethernet cableados. También se deben especificar el nombre y la contraseña del punto de acceso inalámbrico:

```
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: "**********"
```

Conexión a redes inalámbricas WPA Enterprise

También es común encontrar redes inalámbricas protegidas mediante WPA o WPA2 Enterprise, lo que requiere parámetros de autenticación adicionales.

Por ejemplo, si la red está protegida mediante WPA-EAP y TTLS:

```
network:
    version: 2
    wifis:
        wl0:
            access-points:
                workplace:
                    auth:
                        key-management: eap
                        method: ttls
                        anonymous-identity: "@internal.example.com"
                        identity: "joe@internal.example.com"
                        password: "v3ryS3kr1t"
            dhcp4: yes
```

O, si la red está protegida mediante WPA-EAP y TLS:

```
network:
    version: 2
    wifis:
        wl0:
            access-points:
                university:
                    auth:
                        key-management: eap
                        method: tls
                        anonymous-identity: "@cust.example.com"
                        identity: "cert-joe@cust.example.com"
                        ca-certificate: /etc/ssl/cust-cacrt.pem
                        client-certificate: /etc/ssl/cust-crt.pem
                        client-key: /etc/ssl/cust-key.pem
                        client-key-password: "d3cryptPr1v4t3K3y"
            dhcp4: yes
```

Se admiten muchos modos diferentes de cifrado. Consulte la página de referencia de Netplan .

Usando múltiples direcciones en una sola interfaz

La clave de direcciones puede tomar una lista de direcciones para asignar a 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
```

Los alias de interfaz (por ejemplo, eth0: 0) no son compatibles.

Usando múltiples direcciones con múltiples puertas de enlace

Al igual que en el ejemplo anterior, las interfaces con varias direcciones se pueden configurar con varias puertas de enlace.

```
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
```

Dado que hay varias direcciones, cada una con su propia puerta de enlace, no especificamos gateway4aquí y, en su lugar, configuramos rutas individuales a 0.0.0.0/0 (en todas partes) utilizando la dirección de la puerta de enlace para la subred. El metricvalor debe ajustarse para que el enrutamiento se realice como se esperaba.

Se puede utilizar DHCP para recibir una de las direcciones IP de la interfaz. En este caso, la ruta predeterminada para esa dirección se configurará automáticamente con un metricvalor de 100. Como forma abreviada de una entrada inferior routes, gateway4se puede establecer la dirección de puerta de enlace para una de las subredes. En ese caso, se puede omitir la ruta para esa subred routes. Su metric se establece en 100.

Usando Network Manager como renderizador

Netplan admite tanto networkd como Network Manager como backends. Puede especificar qué backend de red se debe usar para configurar dispositivos particulares usando la rendererclave. También puede delegar toda la configuración de la red al propio Network Manager especificando solo la rendererclave:

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

Configurar la vinculación de interfaces

La vinculación se configura declarando una interfaz de vinculación con una lista de interfaces físicas y un modo de vinculación. A continuación, se muestra un ejemplo de un enlace de respaldo activo que usa DHCP para obtener una dirección:

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

A continuación se muestra un ejemplo de un sistema que actúa como un enrutador con varias interfaces vinculadas y diferentes tipos. Tenga en cuenta las declaraciones de clave 'opcional: verdadero' que permiten que se produzca el arranque sin esperar a que esas interfaces se activen por completo.

```
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]
            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
```

Configurar puentes de red

Para crear un puente muy simple que consta de un solo dispositivo que usa DHCP, escriba:

```
network:
    version: 2
    renderer: networkd
    ethernets:
        enp3s0:
            dhcp4: no
    bridges:
        br0:
            dhcp4: yes
            interfaces:
                - enp3s0
```

Un ejemplo más complejo, para que libvirtd use un puente específico con un vlan etiquetado, mientras continúa proporcionando una interfaz sin etiquetar también implicaría:

```
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
```

Entonces libvirtd se configuraría para usar este puente agregando el siguiente contenido a un nuevo archivo XML bajo /etc/libvirtd/qemu/networks/. El nombre del puente en la etiqueta <bridge> y en <nombre> deben coincidir con el nombre del dispositivo de puente configurado mediante netplan:

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

Adjuntar VLAN a interfaces de red

Para configurar varias VLAN con interfaces renombradas:

```
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 ]
```

Llegar a una puerta de enlace conectada directamente

Esto permite configurar una ruta predeterminada, o cualquier ruta, usando la palabra clave “en el enlace” donde la puerta de enlace es una dirección IP que está conectada directamente a la red, incluso si la dirección no coincide con la subred configurada en la interfaz.

```
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
```

Para IPv6, la configuración sería muy similar, con la diferencia notable de un alcance adicional: se requiere vincular la ruta del host a la dirección del enrutador:

```
network:
    version: 2
    renderer: networkd
    ethernets:
        addresses: [ "2001:cafe:face:beef::dead:dead/64" ]
        routes:
            - to: "2001:cafe:face::1/128"
                scope: link
            - to: "::/0"
                via: "2001:cafe:face::1"
                on-link: true
```

Configurar el enrutamiento de origen

Se pueden agregar tablas de ruta a interfaces particulares para permitir el enrutamiento entre dos redes:

En el siguiente ejemplo, ens3 está en la red 192.168.3.0/24 y ens5 está en la red 192.168.5.0/24. Esto permite a los clientes de cualquiera de las redes conectarse a la otra y permitir que la respuesta provenga de la interfaz correcta.

Además, la ruta predeterminada todavía está asignada a ens5, lo que permite que cualquier otro tráfico la atraviese.

```
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
```

Configurar una interfaz de bucle invertido

Networkd no permite la creación de nuevos dispositivos de bucle invertido, pero un usuario puede agregar nuevas direcciones a la interfaz de bucle invertido estándar, lo, para que se considere una dirección válida en la máquina, así como para enrutamiento personalizado:

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

Integración con un servidor DHCP de Windows

Para redes en las que un servidor Windows proporciona DHCP, el uso de la clave dhcp-identifier permite la interoperabilidad:

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

Conexión de un túnel IP

Los túneles permiten a un administrador extender redes a través de Internet configurando dos puntos finales que conectarán una interfaz de túnel especial y realizarán el enrutamiento requerido. Netplan admite túneles SIT, GRE, IP-in-IP (ipip, ipip6, ip6ip6), IP6GRE, VTI y VTI6.

Un uso común de los túneles es habilitar la conectividad IPv6 en redes que solo admiten IPv4. El siguiente ejemplo muestra cómo se puede configurar un túnel de este tipo.

Aquí, 1.1.1.1 es la propia dirección IP del cliente; 2.2.2.2 es la dirección IPv4 del servidor remoto, “2001: dead: beef :: 2/64” es la dirección IPv6 del cliente según lo definido por el túnel, y “2001: dead: beef :: 1” es la dirección IPv6 del servidor remoto .

Finalmente, “2001: cafe: face :: 1/64” es una dirección para el cliente dentro del prefijo IPv6 enrutado:

```
network:
    version: 2
    ethernets:
        eth0:
            addresses:
                - 1.1.1.1/24
                - "2001:cafe:face::1/64"
            gateway4: 1.1.1.254
    tunnels:
        he-ipv6:
            mode: sit
            remote: 2.2.2.2
            local: 1.1.1.1
            addresses:
                - "2001:dead:beef::2/64"
            gateway6: "2001:dead:beef::1"
```

Configuración de funciones virtuales SR-IOV

Para las tarjetas de red SR-IOV, es posible asignar dinámicamente interfaces de función virtual para cada función física configurada. En netplan, un VF se define por tener un vínculo: propiedad que apunta al PF principal.

```
network:
    version: 2
    ethernets:
        eno1:
            mtu: 9000
        enp1s16f1:
            link: eno1
```

```
            addresses : [ "10.15.98.25/24" ]
        vf1:
            match:
                name: enp1s16f[2-3]
            link: eno1
            addresses : [ "10.15.99.25/24" ]
```