Tres herramientas de gestión de redes para gestionarlas a todas

Fernando F. Mancera <ffmancera@riseup.net>
@ffmancera

esLibre

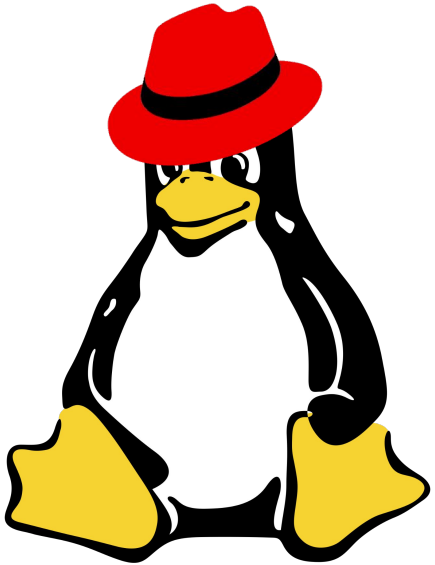
Red Hat

Sobre mi

1. Software Engineer at Red Hat

sugus

netfilter
firewalling, NAT, and packet mangling for linux



esLibre

Red Hat

Objetivo

1. Comprender que es NetworkManager
2. La relación que existe entre NM y los diferentes clientes
3. Saber usar la herramienta adecuada en el momento adecuado

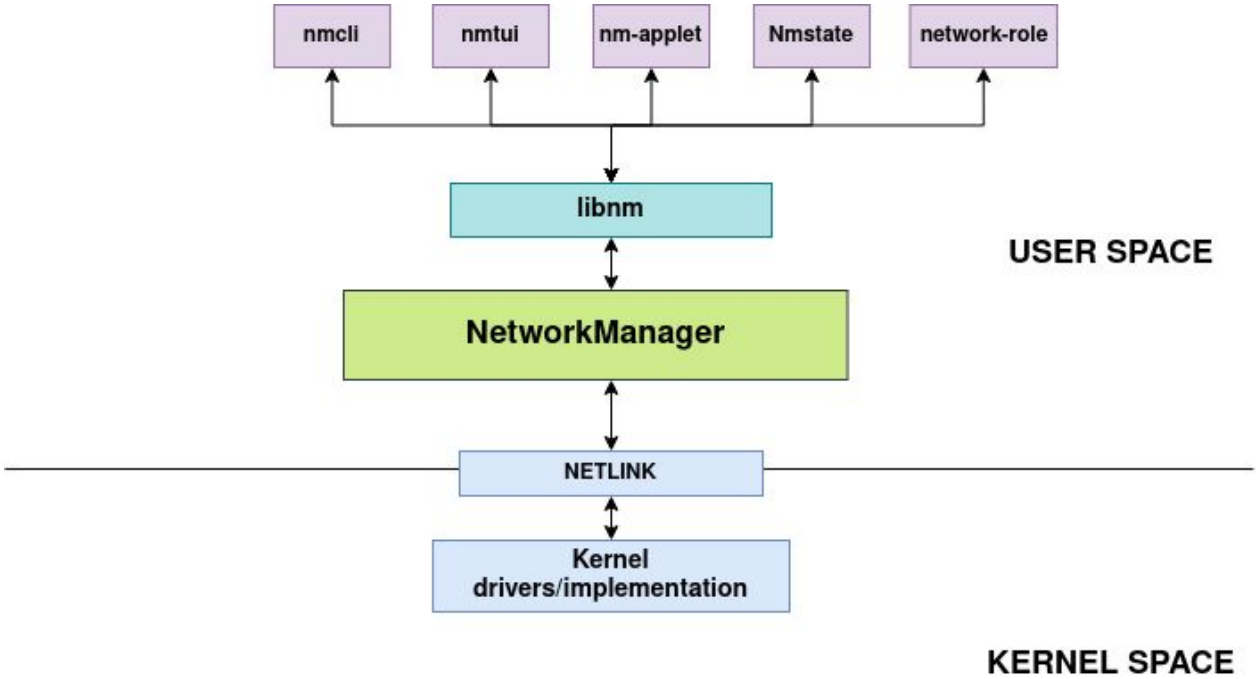
NetworkManager

“NetworkManager is a program for providing detection and configuration for systems to automatically connect to networks. NetworkManager's functionality can be useful for both wireless and wired networks.”

- Archwiki - NetworkManager (https://wiki.archlinux.org/title/NetworkManager)

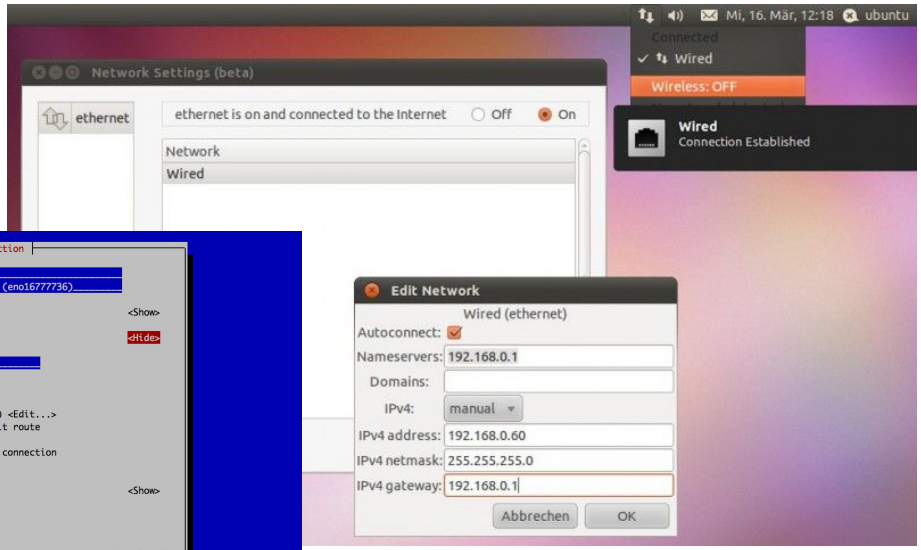
NetworkManager 101

- NetworkManager es un servicio
- Cada NIC tiene un “device” asociado
- El usuario crea configuraciones llamadas “connections”
- Cada “device” puede tener múltiples “connections” asociadas


nmcli
nmtui
nm-applet
Nmstate
network-role
libnm
USER SPACE
NetworkManager
NETLINK
Kernel
drivers/implementation
KERNEL SPACE

Escritorios

- nm-applet
- nmtui



Servidores/entornos de consola

- nmcli
- Nmstate

nmcli connection add type veth con-name veth1 ifname veth1 veth.peer veth2 ipv4.method manual ipv4.dns 8.8.8.8 ipv4.addresses "192.168.100.25/24" connection.autoconnect yes

nmcli connection add type veth con-name veth2 ifname veth2 veth.peer veth1 ipv4.method manual ipv4.dns 8.8.8.8 ipv4.addresses "192.168.100.26/24" connection.autoconnect yes

Nmstate

- Configuración declarativa
- Herramienta de línea de comandos
- Libreria de rust nativa (libnmstate)
- Bindings para C/Python/Golang
- YAML o JSON
- Verificación y rollback
- Se centra en “devices”

```yaml
---
interfaces:
- name: veth1
  state: up
  type: veth
  veth:
    peer: veth2
```

libnmstate

```
#! /bin/python3

import libmstate
from libnmstate.schema import Interface
from libnmstate.schema import InterfaceState
from libnmstate.schema import InterfaceType
from libnmstate.schema import Veth

desired_state = {
        Interface.KEY: [
                {
                        Interface.NAME: "veth1",
                        Interface.STATE: InterfaceState.UP,
                        Interface.TYPE: InterfaceType.VETH,
                        Veth.CONFIG_SUBTREE: {
                                Veth.PEER: "veth2"
                        },
                },
        ],
}

libnmstate.apply(desired_state)
```

Clusters/VMs

- Ansible Linux Network Role
- NMPolicy
- Nmstate

```
- hosts: network-test
  vars:
    network_connections:
      # Specify the bond profile
      - name: bond0
        state: up
        type: bond
        interface_name: bond0
        # ip configuration (optional)
        ip:
          address:
            - "192.0.2.24/24"
            - "2001:db8::23/64"
        # bond configuration settings: (optional)
        bond:
          mode: active-backup
          miimon: 110
```

Ansible Linux Network Role

- Ansible role
- Clusters, VMs.
- NetworkManager y initscripts
- YAML.
- Se centra en “connections”
- Soporte de Nmstate en progreso!

NMPolicy y kubernetes-nmstate

- Kubernetes-nmstate: Configuración de red para entornos con kubernetes
- NMPolicy: Configuración inteligente para clusters con Nmstate

Conclusiones

- Escritorio: nm-applet/nmtui
- Servidor: Nmstate
- Clusters: NMPolicy/Ansible Linux Network Role
- Kubernetes: kubernetes-nmstate

¡Gracias!
¿Alguna pregunta?

Fernando F. Mancera <ffmancera@riseup.net>
@ffmancera

esLibre

Red Hat