Networking management made simple with Nmstate

Taming the internals of NetworkManager

Fernando F. Mancera

Senior Software Engineer

Red Hat

What we'll discuss today

- NetworkManager
- Nmstate, being declarative
- Why netlink and not sysfs?
- Nmstate handles everything
- Let's see examples
- In action!

NetworkManager

Networking that Just Works!

- NetworkManager is the standard Linux network configuration tool suite.
  - There are multiples tools around it like nm-applet, nmtui, nmcli, nm-cloud-setup..
  - The NetworkManager daemon do most of the work when configuring something.

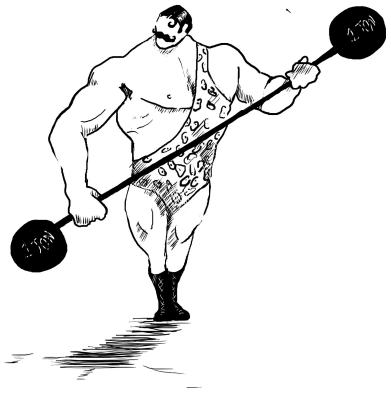
Source:
https://networkmanager.dev/

Nmstate, being declarative

- Nmstate is a library with an accompanying command line tool that manages host networking settings in a declarative manner.
  - It communicates with NetworkManager to configure the network and perform rollback/checkpoint
  - It uses Nispor to communicate with kernel via netlink and fetch real-time kernel networking configuration.

Why netlink and not sysfs?

- Sysfs is not an API. It might break between releases.
- Netlink **IS** an API.
  - Netlink is **STABLE**
  - It is **NOT** deprecated
  - Use **SOCKETS** not files



Source:
https://www.kernel.org/doc/html/latest/admin-guide/sysfs-rules.html

Nmstate handles everything

- It manages the interdependencies of the interfaces
- Does validation, normalization and verification
  - That means it will point you what is going wrong when configuring your networking

Source:

Let’s see examples

```yaml
interfaces:
- name: bond99
  type: bond
  state: up
  ipv4:
    address:
    - ip: 192.0.2.0
      prefix-length: 24
    enabled: true
  link-aggregation:
    mode: balance-rr
    options:
      miimon: '140'
    port:
    - eth3
    - eth2
```

```yaml
---
interfaces:
  - name: eth1.101
    type: vlan
    state: up
    vlan:
      base-iface: eth1
      id: 101
```

```yaml
---
interfaces:
  - name: linux-br0
    type: linux-bridge
    state: up
    bridge:
      options:
        group-forward-mask: 0
        mac-ageing-time: 300
        multicast-snooping: true
        stp:
          enabled: true
          forward-delay: 15
          hello-time: 2
          max-age: 20
          priority: 32768
      port:
        - name: eth1
          stp-hairpin-mode: false
          stp-path-cost: 100
          stp-priority: 32
```


Source:

Let’s see examples

```yaml
---
interfaces:
  - name: eth1
    type: ethernet
    state: up
    ipv4:
      address:
      - ip: 192.0.2.251
        prefix-length: 24
      dhcp: false
      enabled: true

routes:
  config:
  - destination: 198.51.100.0/24
    metric: 150
    next-hop-address: 192.0.2.1
    next-hop-interface: eth1
    table-id: 254
```

```yaml
route-rules:
  config:
    - ip-to: 192.0.2.0/24
      ip-from: 198.51.100.0/24
      priority: 100
      route-table: 254
      fwmark: 0x30
      fwmask: 0x10
```

```yaml
---
dns-resolver:
  config:
    search:
    - example.com
    - example.org
    server:
    - 2001:4860:4860::8888
    - 8.8.8.8
```

```yaml
interfaces:
- name: ovs0
  type: ovs-interface
  state: up
  ipv4:
    enabled: true
    address:
      - ip: 192.0.2.1
        prefix-length: 24
- name: ovs-br0
  type: ovs-bridge
  state: up
  bridge:
    options:
      stp: true
    port:
    - name: eth3
    - name: ovs0
```


Source:

In action!

- Demo time, sorry if it doesn't work :-)

Source:

Questions?

Feel free to ask questions! There are not dumb questions :-)

Contact me at ffmancera@riseup.net or ffmancera@mastodon.social

Red Hat