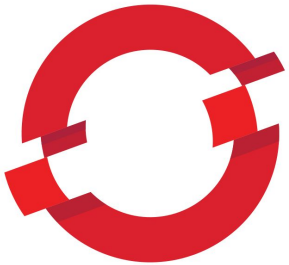
Declare Your Linux Network State!

with nmstate

Edward Haas, Red Hat <edwardh@redhat.com>
Till Maas, Red Hat <till@redhat.com>

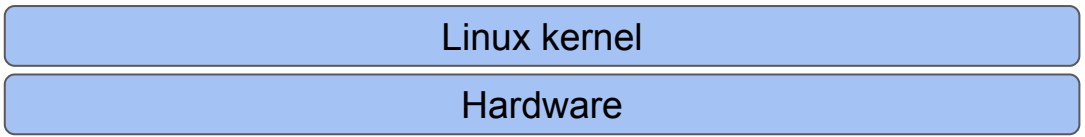
ANSIBLE

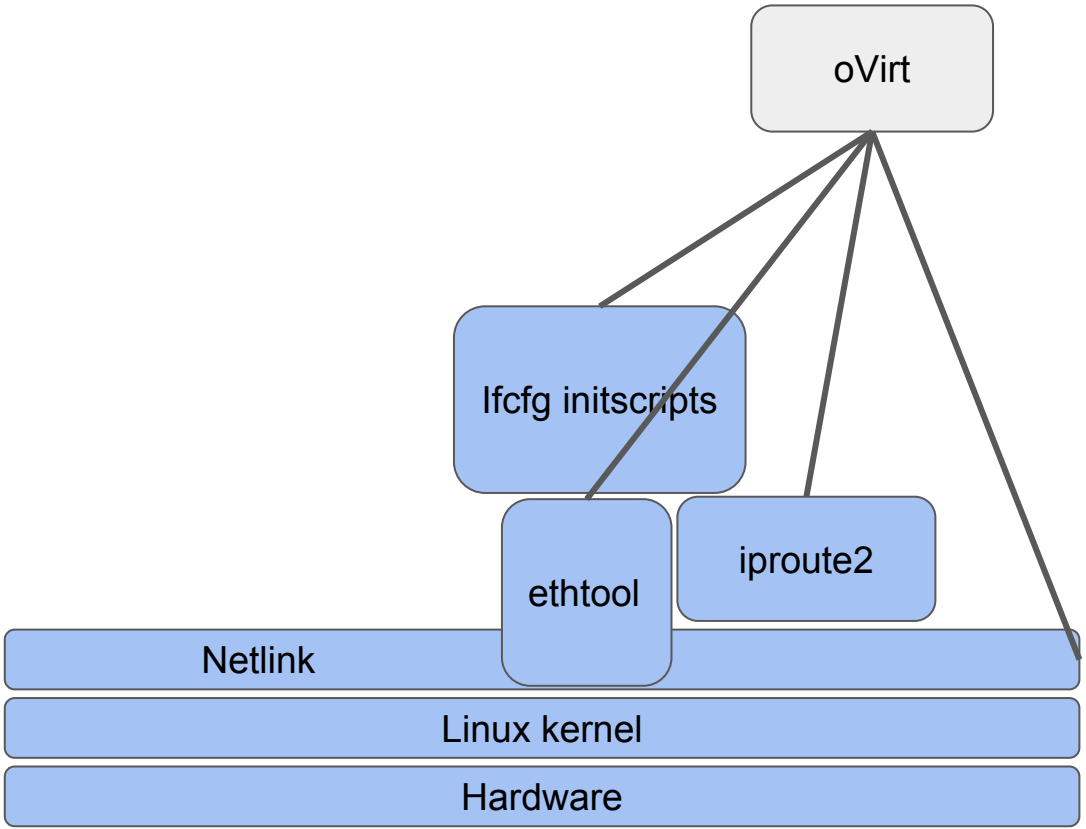
oVirt

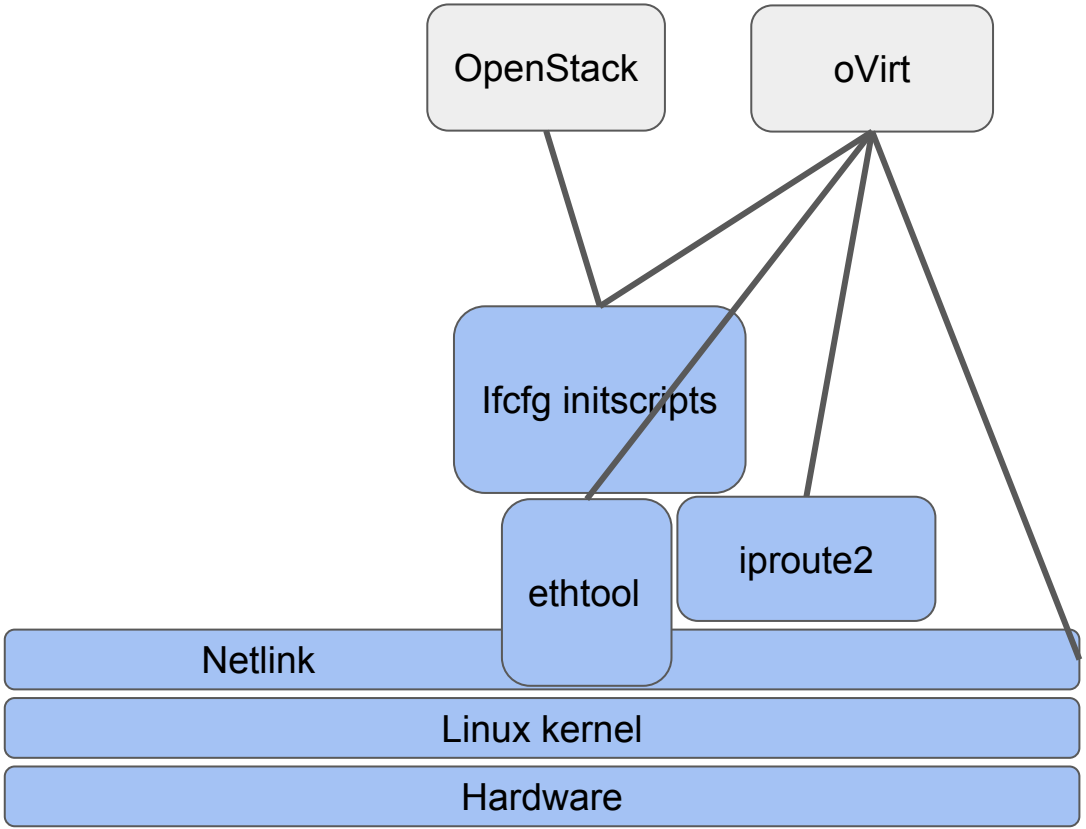


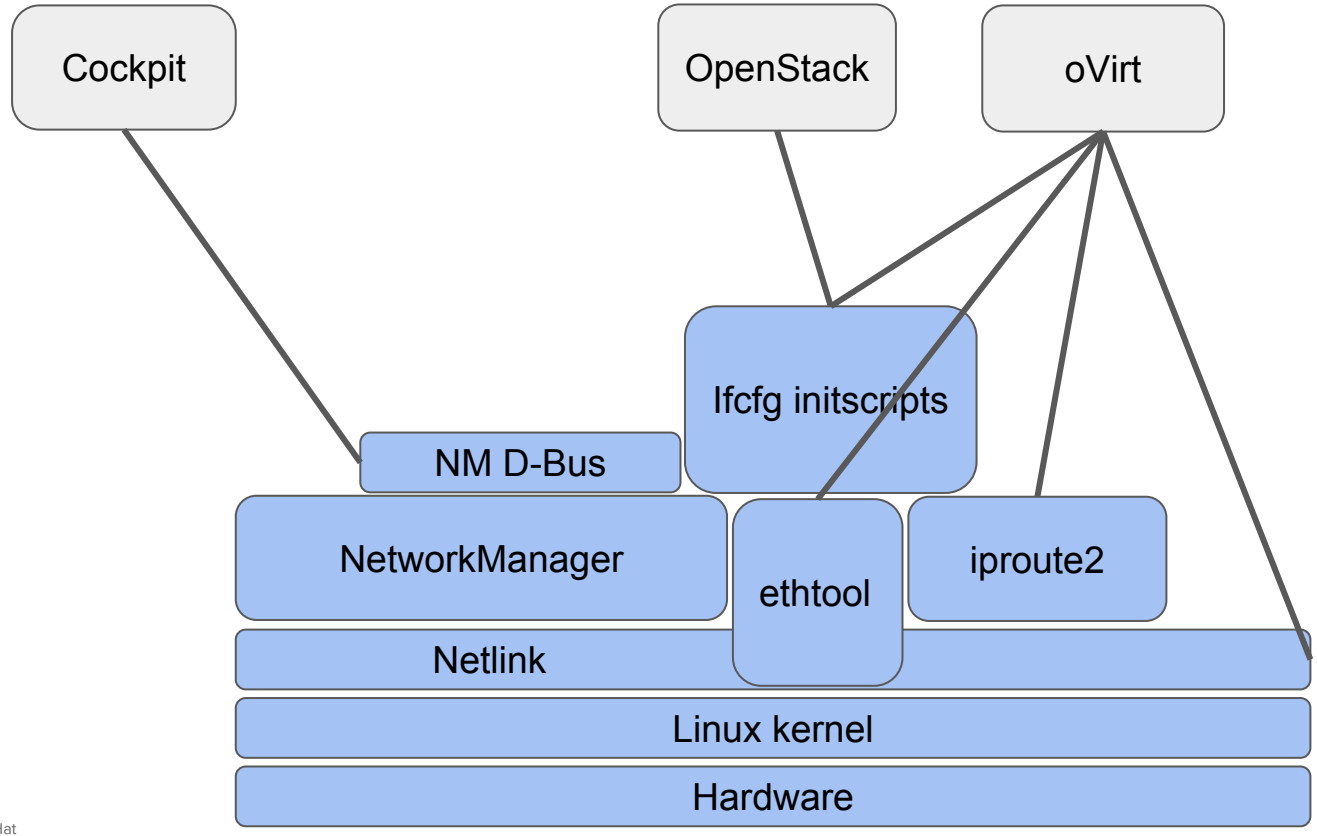
OPENSHIFT

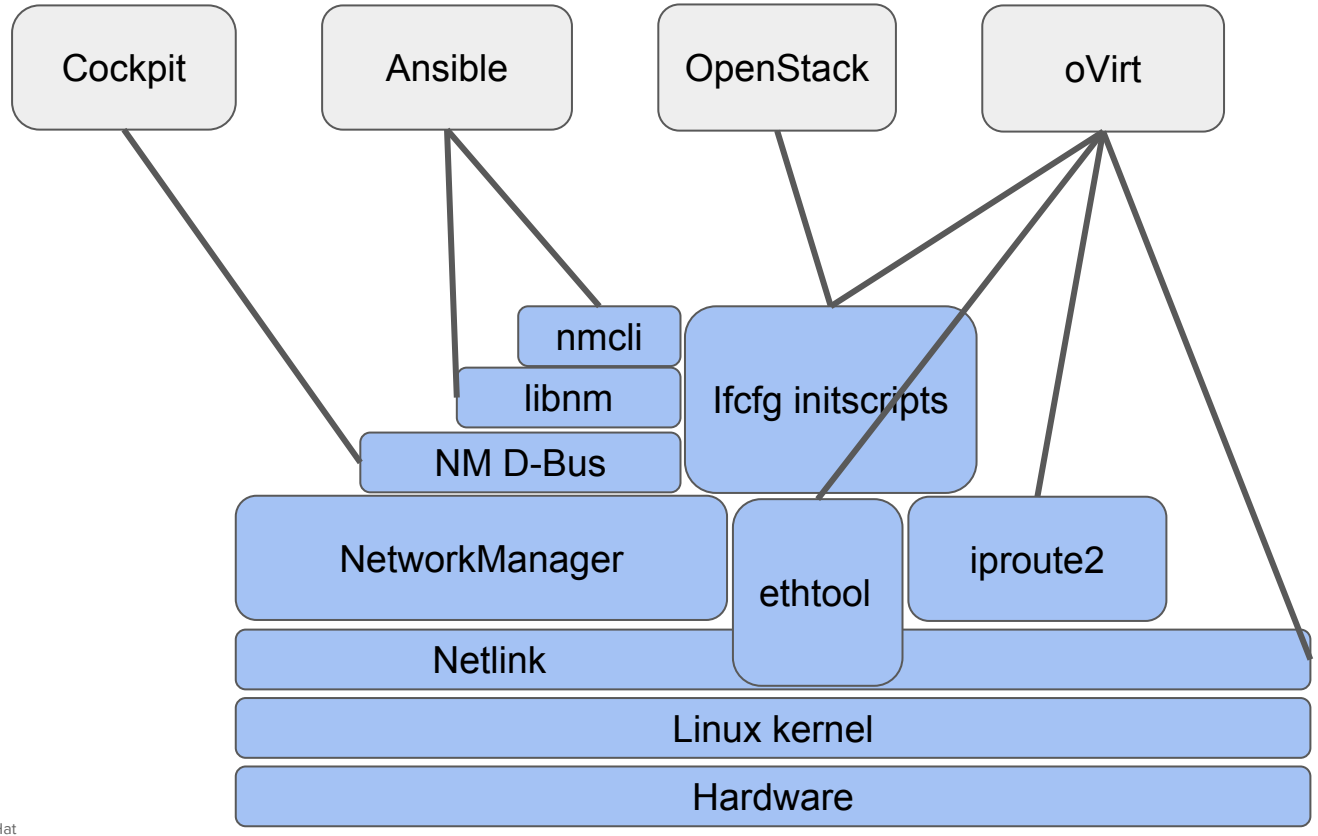
openstack®

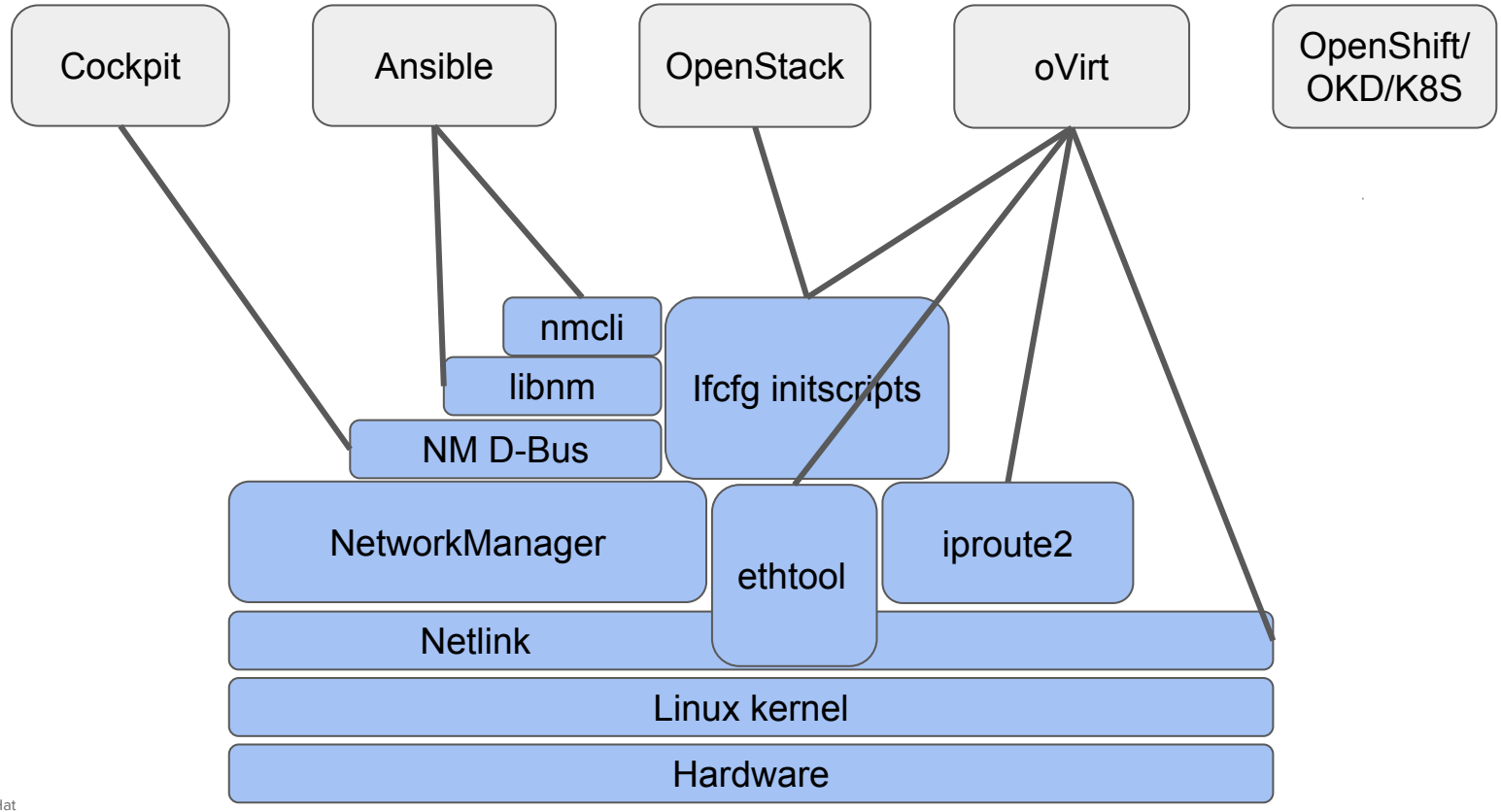

Linux kernel
Hardware

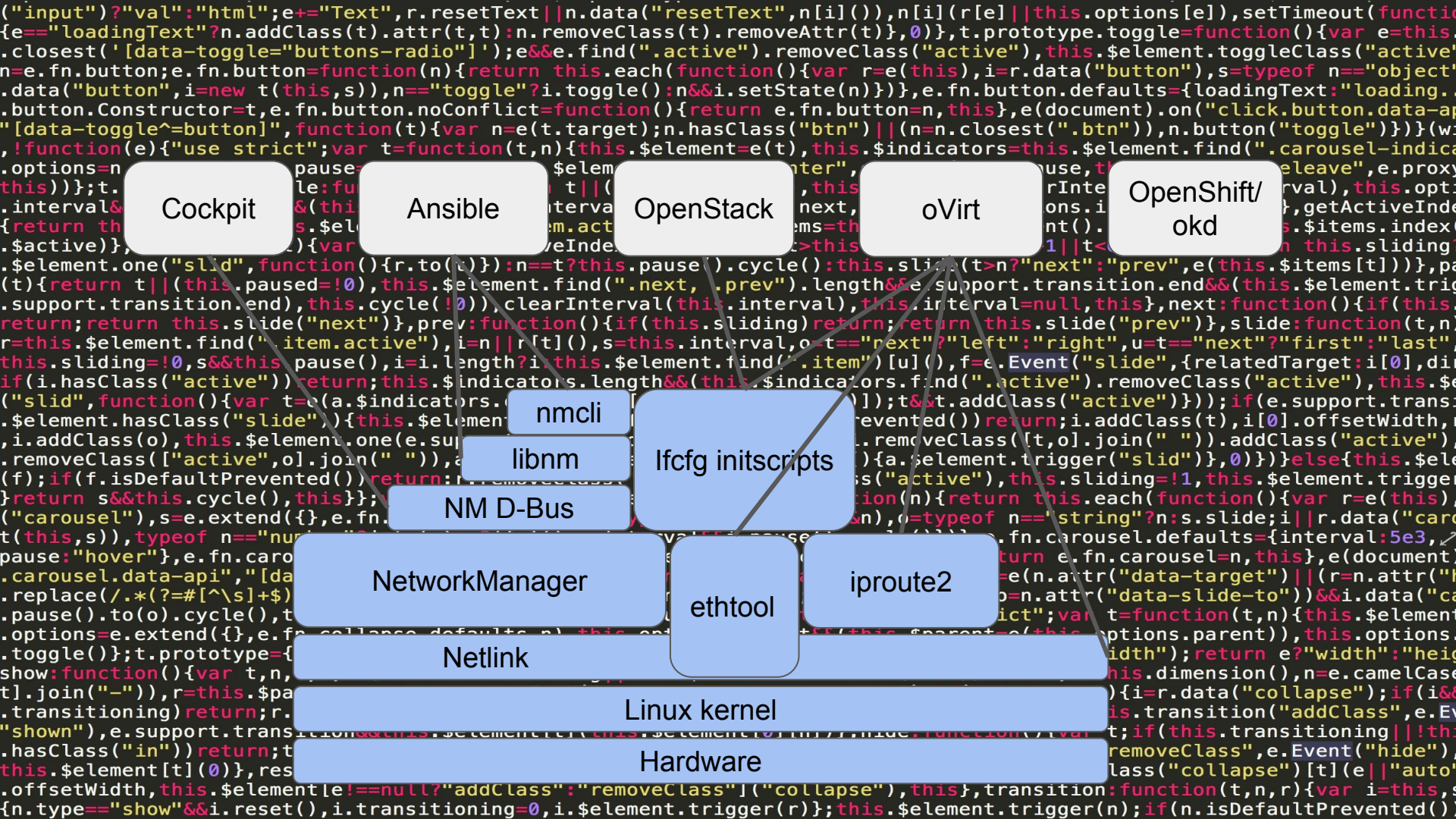

oVirt
Ifcfg initscripts
ethtool
iproute2
Netlink
Linux kernel
Hardware


OpenStack
oVirt
Ifcfg initscripts
ethtool
iproute2
Netlink
Linux kernel
Hardware


Cockpit
OpenStack
oVirt
Ifcfg initscripts
NM D-Bus
NetworkManager
ethtool
iproute2
Netlink
Linux kernel
Hardware

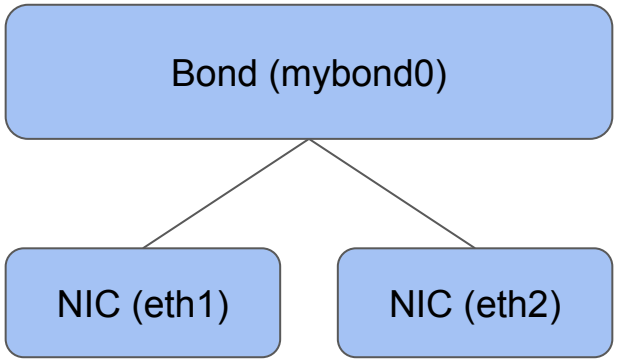

Cockpit
Ansible
OpenStack
oVirt
nmcli
libnm
NM D-Bus
Ifcfg initscripts
NetworkManager
ethtool
iproute2
Netlink
Linux kernel
Hardware


Cockpit
Ansible
OpenStack
oVirt
OpenShift/
OKD/K8S
nmcli
libnm
NM D-Bus
Ifcfg initscripts
NetworkManager
ethtool
iproute2
Netlink
Linux kernel
Hardware


Cockpit
Ansible
OpenStack
oVirt
OpenShift/
okd
nmcli
libnm
Ifcfg initscripts
NM D-Bus
NetworkManager
ethtool
iproute2
Netlink
Linux kernel
Hardware

Configure a Bond

Intro Example



Configure a Bond (nmcli)

Intro Example

$ nmcli con add type bond ifname mybond0 mode active-backup

$ nmcli con add type ethernet ifname eth1 master mybond0

$ nmcli con add type ethernet ifname eth2 master mybond0

$ nmcli con mod mybond0 ipv4.method manual ipv4.address "1.2.3.4/24"

Configure a Bond (iproute2)


```
# ip link add mybond0 type bond

# ip link set eth1 master mybond0

# ip link set eth2 master mybond0

# ip addr add 1.2.3.4/24 dev mybond0
```

Configure a Bond (ifcfg)

Intro Example

```
DEVICE=mybond0
mode=active-backup
TYPE=Bond
BONDING_MASTER=yes
IPV6INIT=no
NAME=bond00
ONBOOT=yes
BOOTPROTO=none
IPADDR=1.2.3.4
PREFIX=24
DEFROUTE=yes
```

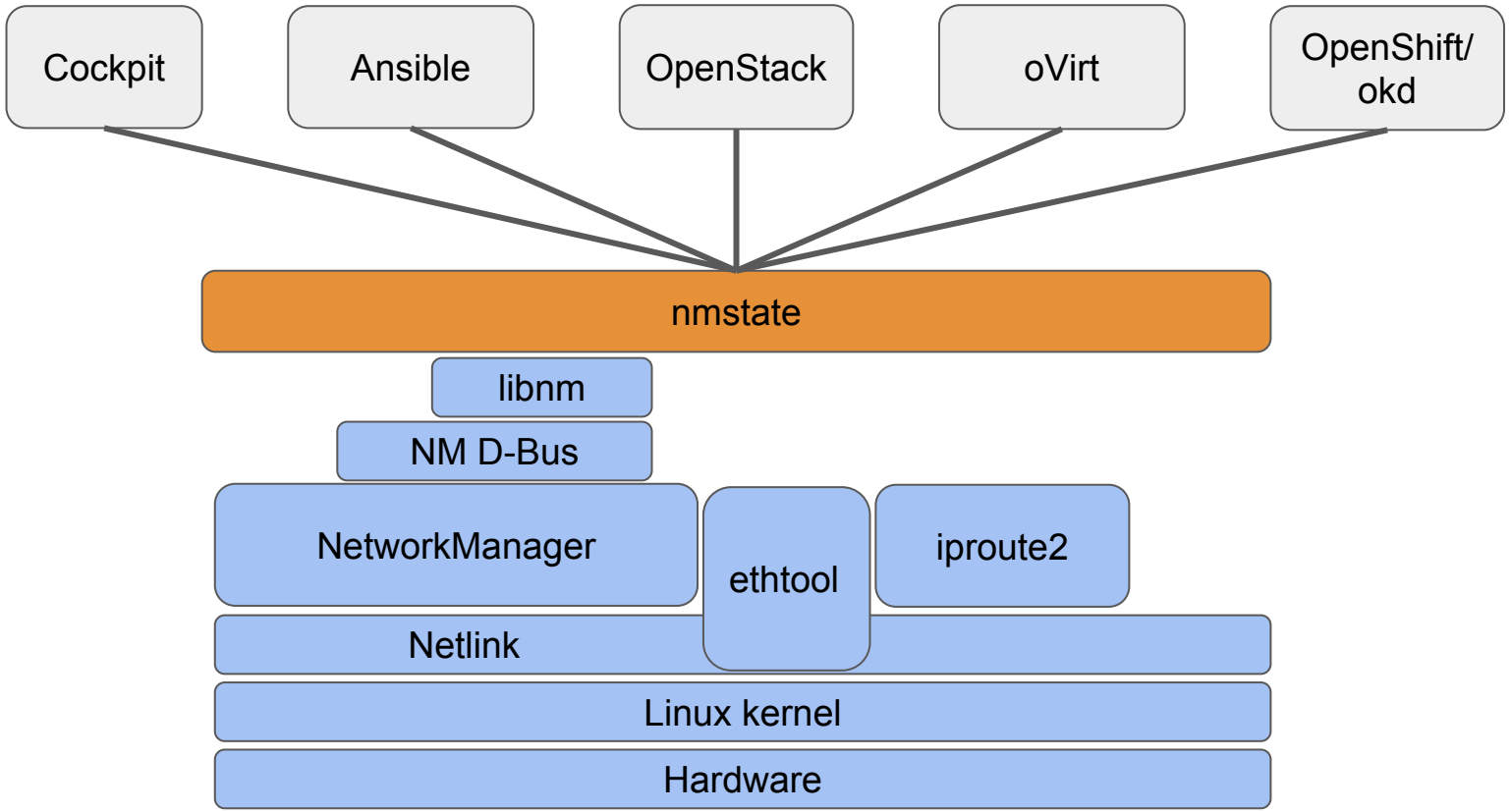
```
TYPE=Ethernet
NAME=eth1
DEVICE=eth1
ONBOOT=yes
MASTER=mybond0
SLAVE=yes
```

```
TYPE=Ethernet
NAME=eth2
DEVICE=eth2
ONBOOT=yes
MASTER=mybond0
SLAVE=yes
```

Configure a Bond (nmstate)

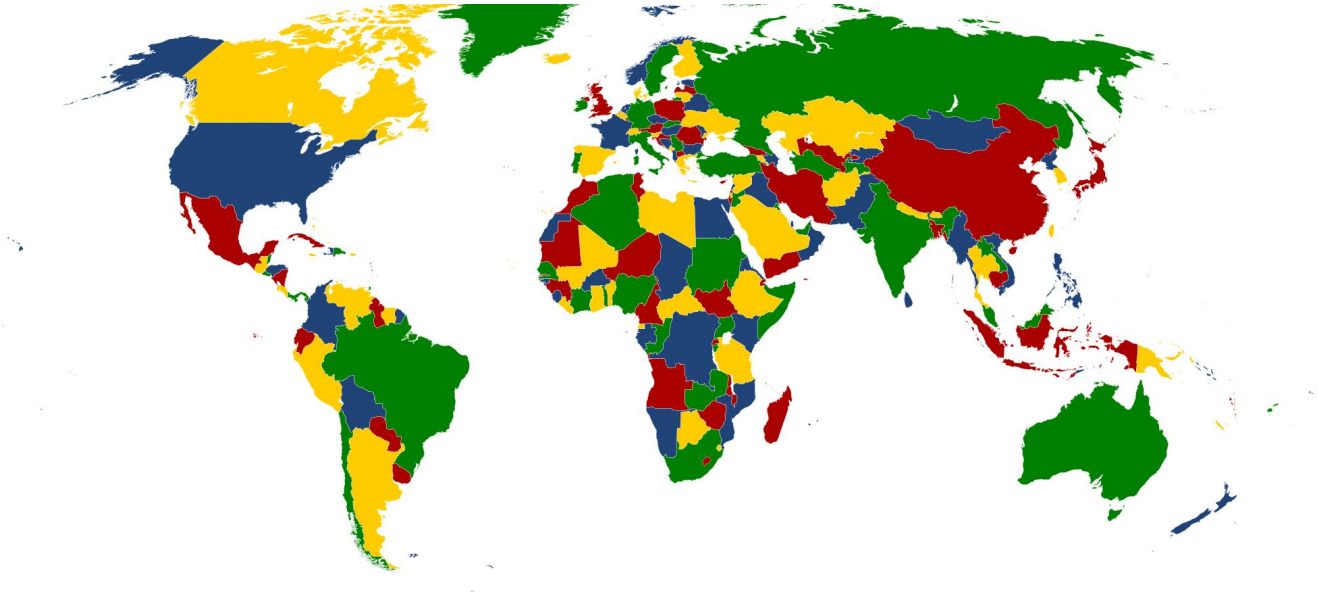
Intro Example

```
interfaces:
- name: mybond0
  type: bond
  state: up
  link-aggregation:
    mode: active-backup
    slaves:
    - eth1
    - eth2
  ipv4:
    enabled: true
    address:
    - ip: 1.2.3.4
      prefix-length: 24
```


Cockpit
Ansible
OpenStack
oVirt
OpenShift/
okd
nmstate
libnm
NM D-Bus
NetworkManager
ethtool
iproute2
Netlink
Linux kernel
Hardware

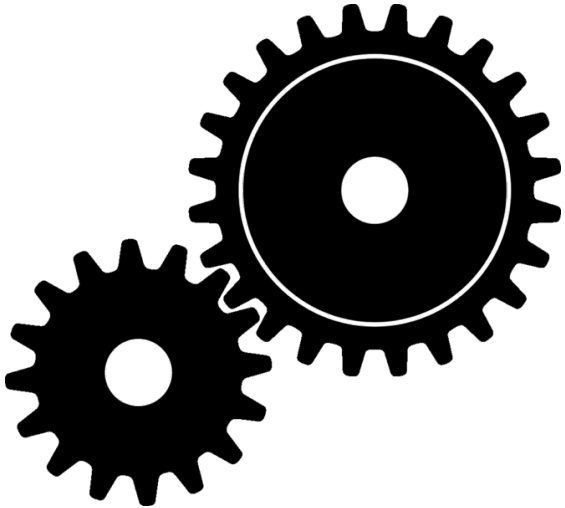
Design

Complete Linux host network state



Design

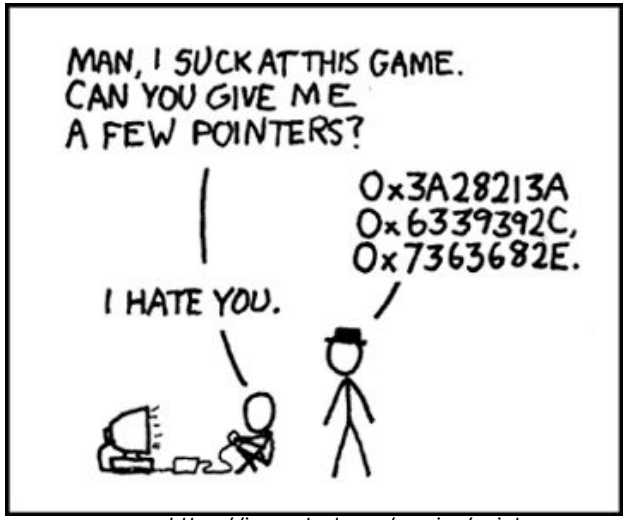
Configuration **and** reporting



Design

Declarative

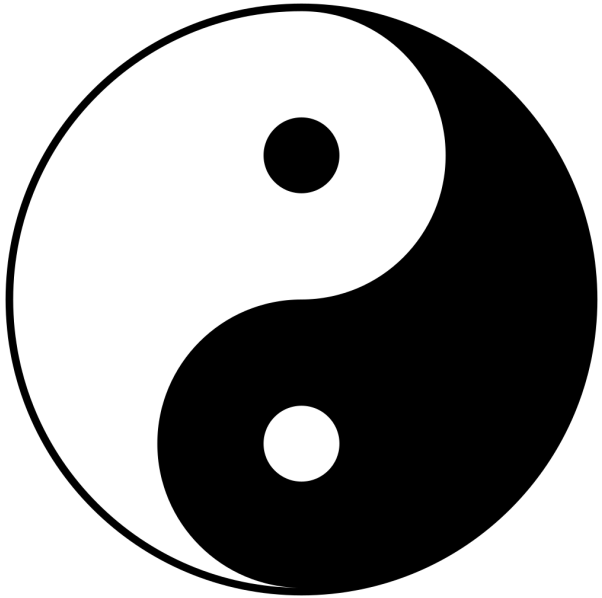
Describe WHAT you want and don’t bother with the HOW.



https://imgs.xkcd.com/comics/pointers.png

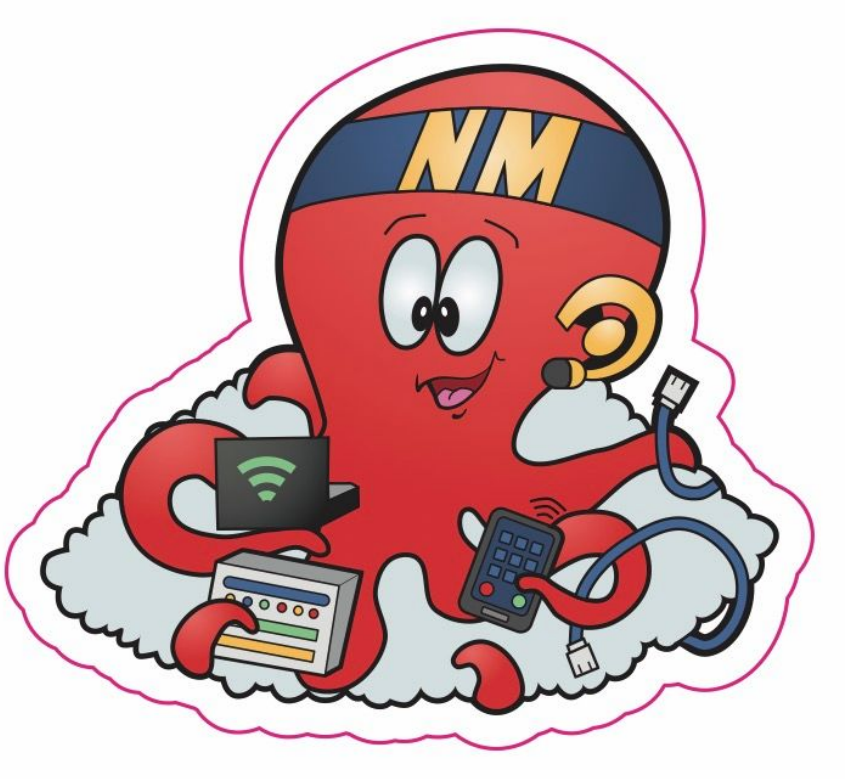
Design

Inspired by IETF Network Modeling Working Group (NETCONF/YANG)



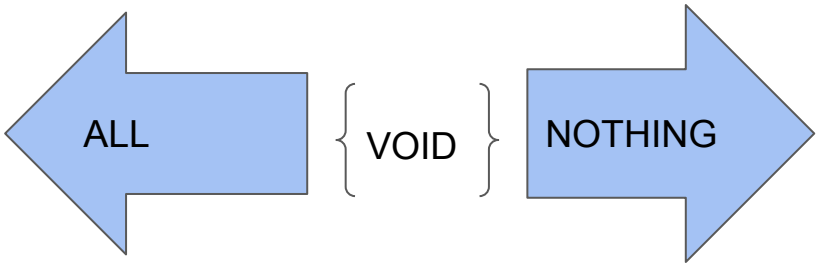
Design

- Based on NetworkManager
- Open for provider extensions



Design

Atomic changes



Design

Allow partial states

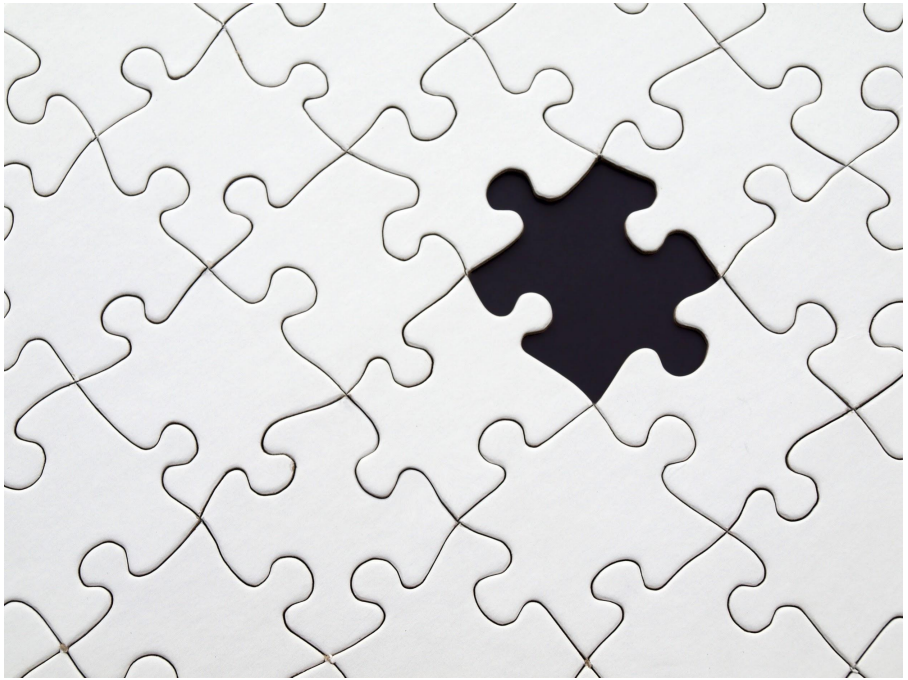
https://commons.wikimedia.org/wiki/File:Partial_Eclipse_of_Moon.jpg

Nmstate Design

- simple API:
  - state = show()
  - apply(state)
- allow partial states to configure only subset of all settings
- verification of the configuration by comparing the runtime state
- atomic configuration changes: Rollback to previous state on failure by default
- Use power of NetworkManager but allow enhancements missing in NetworkManager

Currently support devices

- Ethernet
- IPv4 & IPv6, static & dynamic
- Bonding
- Linux bridges
- OVS bridges (basic)



Command-line interface

```
# nmstatectl  show eth0
---
interfaces:
- name: eth0
  type: ethernet
  state: up
  mtu: 1500
  ipv4:
    enabled: true
    dhcp: true
    address:
    - ip: 192.168.122.197
      prefix-length: 24
```

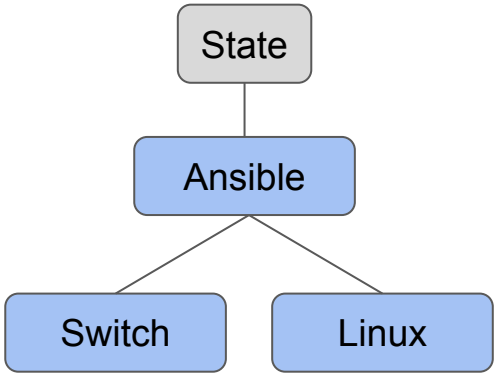
Simple (Python) API

```
state = netinfo.show()
state[‘interfaces’][0][‘mtu’] = 9000
netapplier.apply(state)
```

Verification with rollback

Ansible network modules

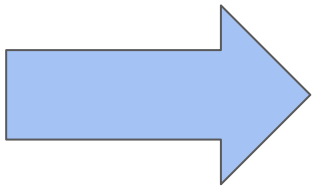
- net_interface
- net_linkagg
- net_vlan
- net_l3_interface



Ansible network modules for Linux

```
# Ansible

tasks:
  - net_linkagg:
      name: web-bond
      state: up
      members:
        - eth1
        - eth2
```



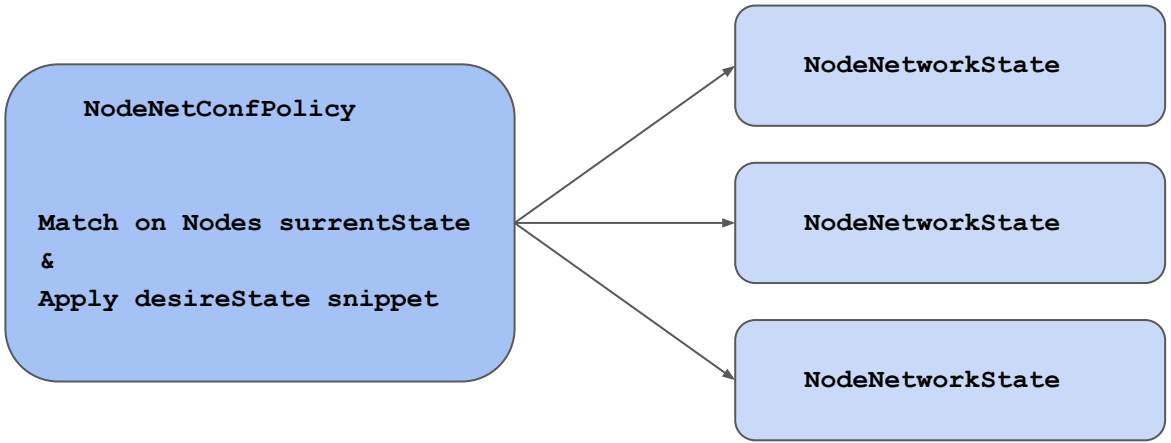
```
# nmstate

---
interfaces:
- name: web-bond
  type: bond
  state: up
  link-aggregation:
    mode: 802.3ad
    options: {}
    slaves:
    - eth1
    - eth2
```

kubernetes-nmstate (PoC)

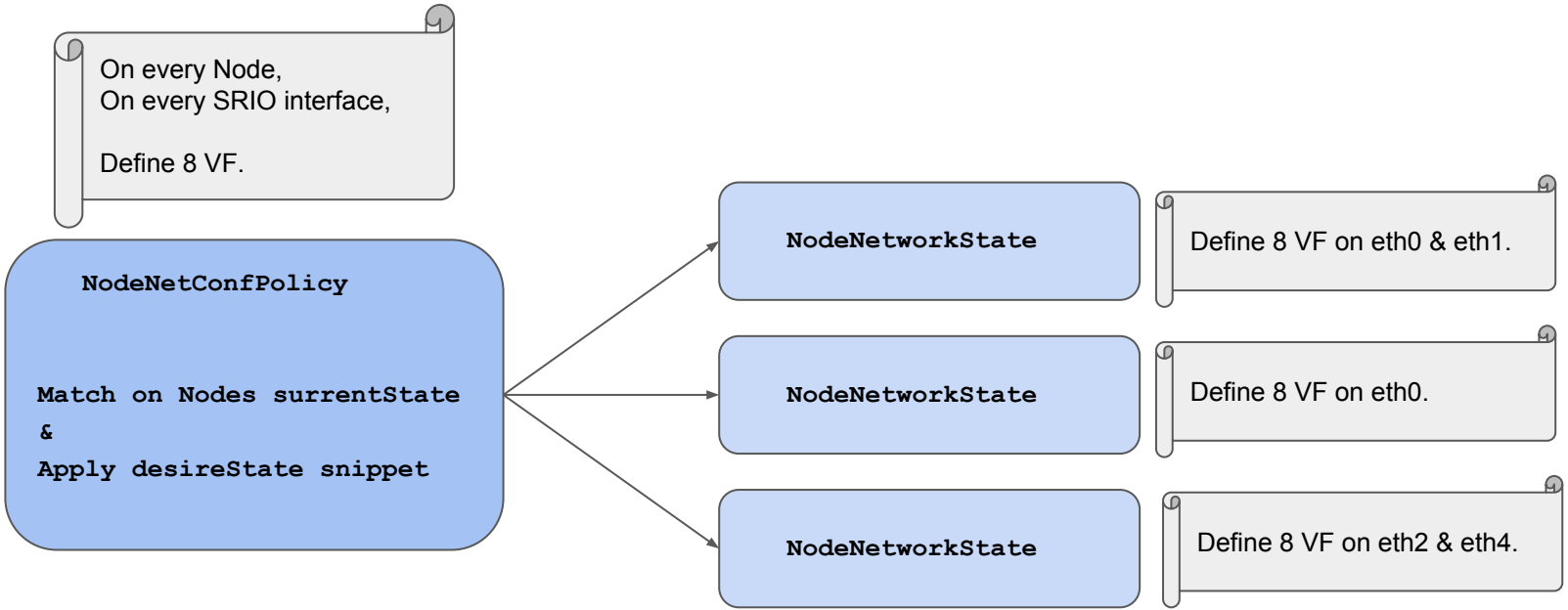
- Manage host/node network through Kubernetes.

- Implements the suggested Kubernetes Node Network Configuration CRD



PoC: https://github.com/nmstate/kubernetes-nmstate

kubernetes-nmstate (PoC)



kubernetes-nmstate (PoC)

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NodeNetworkState
metadata:
  name: my-node-netstate
spec:
  managed: true
  nodeName: my-node
  desiredState:
    interfaces:
     - name: bond0
        type: bond
        state: up
        link-aggregation:
          mode: balance-rr
          slaves:
            - eth0
            - eth1
        ipv4:
          enabled: true
          address:
            - ip: 10.10.10.2
              prefix-length: 24
        ipv6:
          enabled: false
```

```
status:
  currentState:
    capabilities: []
    interfaces:
      - if-index: 10
        name: bond0
        type: bond
        state: up
        phys-address: aa:bb:cc:dd:ee:ff
        link-aggregation:
          mode: balance-rr
          slaves:
            - eth0
            - eth1
        ipv4:
          enabled: true
          address:
            - ip: 10.10.10.2
              prefix-length: 24
        ipv6:
          enabled: false
```

Challenges

```
# desired state

---
interfaces:
- name: eth0
  type: ethernet
  state: up
  ipv4:
    enabled: true
    dhcp: true
```

```
# actual state

---
interfaces:
- name: eth0
  type: ethernet
  state: up
  ipv4:
    enabled: true
    dhcp: true
    address:
    - ip: 192.168.122.197
      prefix-length: 24
```


nmstate.io

How to participate

Development: https://github.com/nmstate/nmstate

Planning: https://nmstate.atlassian.net

Discussions:

- NetworkManager mailing list
- #nmstate on Freenode IRC

Outlook

- Different state for configuration/persistence and runtime
- Support commit and confirm commands
- More interface types
- Add read-only report values
- Proprietary vendor interfaces
- NETCONF/YANG
- Routing (under review)
- Firewall
- Integration with oVirt, OpenStack, KubeVirt

Thank you!

nmstate.io