

# A declarative approach to Linux networking with Netplan

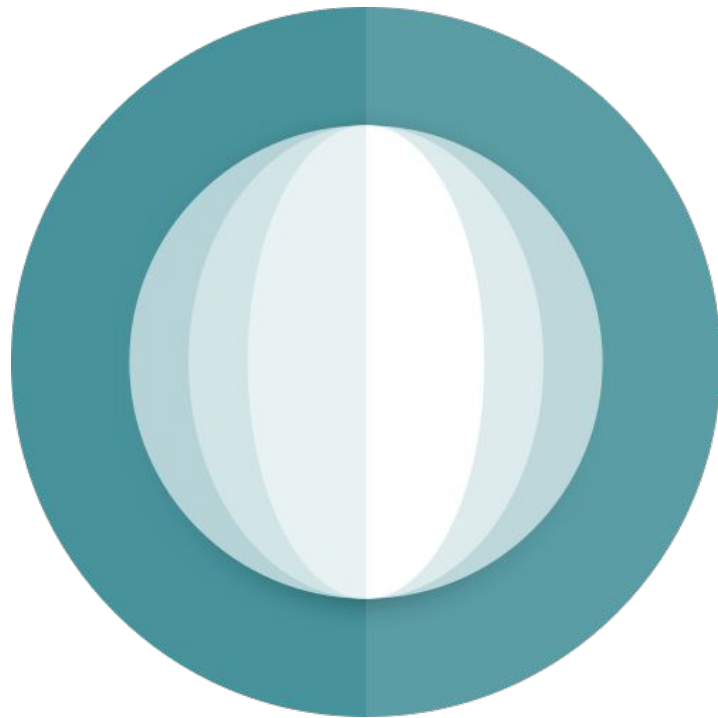
DebConf23, Kochi/India



# Netplan – Declarative network configuration

## Table of contents

- About myself
- What is Netplan?
  - But why?
  - Some History
- Netplan usage
  - Status quo
- Future work
  - QA



Section #0

# About myself

# About myself – Lukas ‘slyon’ Märdian

- Based in Germany
  - First time at DebConf, please be kind ;-)
- Contributor since 10+ years
  - Debian pkg-fso team, Openmoko anyone?
- Ubuntu Foundations
  - At Canonical since 2020
- DM since 2021, DD since 2023



Netplan Maintainer

Ubuntu Core-Dev

Debian Developer

early 2020

late 2020

mid 2023

[slyon@debian.org](mailto:slyon@debian.org) | [slyon@ubuntu.com](mailto:slyon@ubuntu.com)

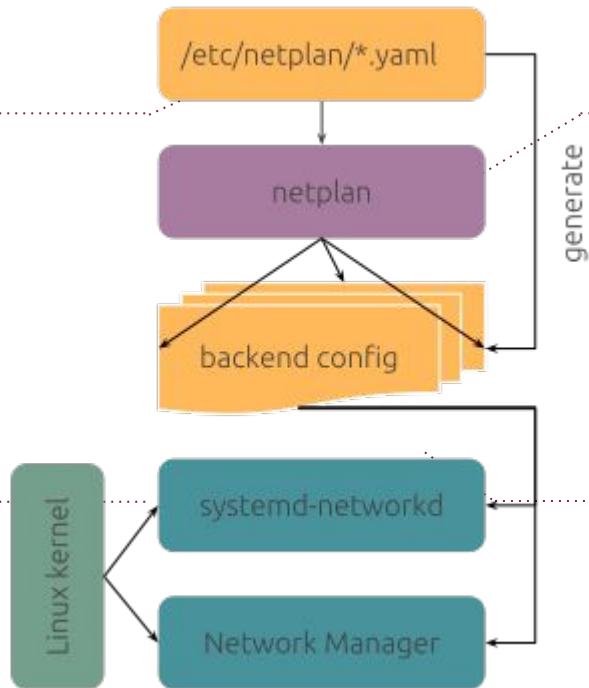
Section #1

# What is Netplan?

# What is Netplan?

Common interface for  
networking daemons  
**YAML Frontend**

Some sugar on top:  
SR-IOV, OVS, regdom,  
**Interactive CLI**



Implemented as  
systemd-generator  
**Slim one-off binary**

systemd-networkd &  
NetworkManager  
**Backend support**

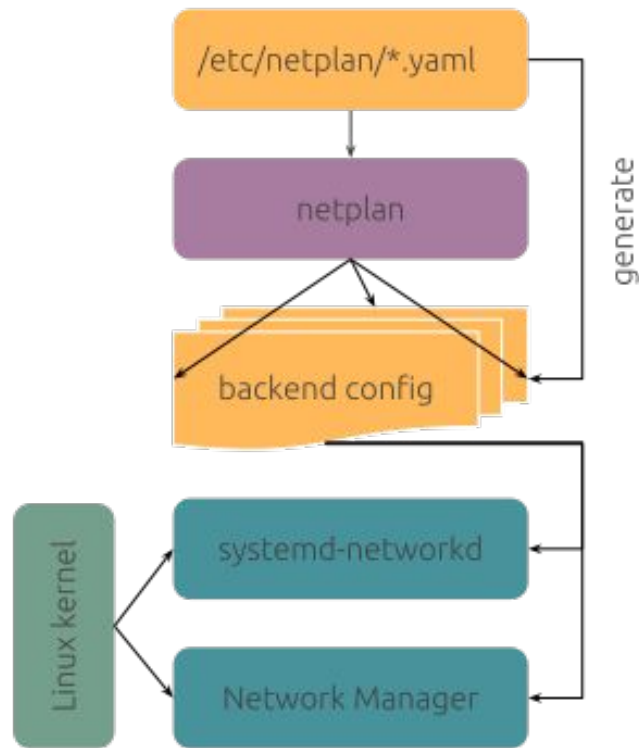
# What is Netplan?

## Declarative network config

Netplan reads network configuration from `/etc/netplan/*.yaml` written by admins, installers, cloud image metadata or other OS deployments.

During early boot, Netplan generates backend specific configuration files in `/run` to hand off control of interfaces to a particular backend:

- `systemd-networkd`
- `NetworkManager`



Section #1.1

# But ...why?



# Why Netplan?

Keep simple things simple, but allow for custom setups of any complexity.

IPv6, bridges, bonds, ifupdown, VRFs, VXLANs, ifupdown-ng, tunnels, declarative, iproute2, **NetworkManager**, access-points, dummy, routing policy, veth, WWAN, ifupdown2, VLANs, ethernet, imperative, InfiniBand, DNS, **systemd-networkd**, hardware offloading, DHCP, sysctl, router-advertisement


- Network ecosystem is scattered
- Active upstream & feature development
- Modern codebase
- Test automation
- Combining the best of two worlds



# Why Netplan?


Keep simple things simple, but allow for custom setups of any complexity.































- **systemd-networkd**
  - Slim, static, text-based configuration
  - Cloud-init integration
- **NetworkManager**
  - Interactive UI, dynamic WiFi, cellular
  - Desktop integration
- Unify config file path
- Good test automation
- Detailed documentation, netplan(5), [netplan.rtfd.io](https://netplan.rtfd.io)


Debian Continuous Integration
[Home](#)
[Status](#)
[Documentation](#)

n / netplan.io

netplan.io


[Tracker](#)
[Excuses](#)
[Bug Reports](#)

	unstable	testing	stable	oldstable
amd64	 0.107-4 pass	 0.107-2 pass	 0.106-2 pass	 fail
arm64	 0.107-4 pass	 0.107-2 pass	 0.106-2 pass	 fail
armel	 0.107-4 pass	 0.107-2 pass	 0.106-2 pass	 fail
armhf	 0.107-4 pass	 0.107-2 pass	 0.106-2 pass	 fail
i386	 0.107-4 pass	 0.107-2 pass	 0.106-2 pass	 fail
ppc64el	 0.106.1-8 pass	 0.107-2 pass	 0.106-2 pass	 fail
riscv64	 0.107-3 pass	 n/a tmpfail	? No test data	? No test data
s390x	 0.107-3 pass	 0.107-2 pass	 0.106-2 pass	 fail

Section #1.2

# Some History

# Some History

## Ubuntu's default networking tool

- Default Ubuntu networking **since 2016**
  - 18.04 LTS, 20.04 LTS & 22.04 LTS ...
  - Dependency based boot
  - Unify config files across server & desktop
  - Unify cloud metadata and & installers
- Used by millions of users
- Available on AWS, GCP and other public clouds, through cloud-init
- Available on Debian, Arch Linux, Fedora, EPEL/RHEL, Ubuntu, ...



Section #2

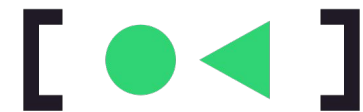
# Netplan usage

# Netplan usage: simple

## Simple example & drop-in configuration

/etc/netplan/30-user.yaml

```
network:  
  version: 2  
  ethernets:  
    enp3s0:  
      dhcp4: true  
      dhcp6: true
```



# systemd

**Output generated:**

/run/systemd/network/10-netplan-enp3s0.network

# Netplan usage: simple

## Simple example & drop-in configuration

/etc/netplan/30-user.yaml

```
network:
  version: 2
  ethernets:
    enp3s0:
      dhcp4: true
      dhcp6: true
```

/usr/lib/netplan/00-nm-all.yaml

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



### Output generated:

~~/run/systemd/network/10-netplan-enp3s0.network~~

/run/NetworkManager/system-connections/  
netplan-enp3s0.nmconnection

# Netplan usage: complex

## Open vSwitch example

```
network:
  version: 2
  ethernets:
    eth0: {}
    eth1: {}
  bonds:
    ovsbond:
      interfaces: [eth0, eth1]
      openvswitch:
        lacp: active
  bridges:
    ovsbr:
      addresses: [192.170.1.1/24]
      interfaces: [ovsbond]
```

```
examples > ! openvswitch.yaml
1 network:
2   version: 2
3   openvswitch:
4     protocols: [OpenFlow13, OpenFlow14, OpenFlow15]
5     ports:
6       - [patch0-1, patch1-0]
7     ssl:
8       ca-cert: /some/ca-cert.pem
9       certificate: /another/cert.pem
10      private-key: /private/key.pem
11    external-ids:
12      somekey: somevalue
13    other-config:
14      key: value
15  ethernets:
16    eth0:
17      addresses: [10.5.32.26/20]
18      openvswitch:
19        external-ids:
20          iface-id: mylocaliface
21        other-config:
22          disable-in-band: false
23    eth1: {}
24  bonds:
25    bond0:
26      interfaces: [patch1-0, eth1]
27      openvswitch:
28        lacp: passive
29      parameters:
30        mode: balance-tcp
31  bridges:
32    ovs0:
33      addresses: [10.5.48.11/20]
34      interfaces: [patch0-1, eth0, bond0]
35      openvswitch:
36        protocols: [OpenFlow10, OpenFlow11, OpenFlow12]
37        controller:
38          addresses: [unix:/var/run/openvswitch/ovs0.mgmt]
39          connection-mode: out-of-band
40        fail-mode: secure
41        mcast-snooping: true
42      external-ids:
43        iface-id: myhostname
44      other-config:
45        disable-in-band: true
```



# Netplan usage: CLI & API

## Interactive runtime configuration

- Netplan CLI
  - Written in Python (optional)
  - “netplan apply/try”
  - “netplan get/set”
  - “netplan status”
- Making use of libnetplan.so
  - YAML parser
  - Logical validation of configuration
  - Python bindings available in **python3-netplan**

```
lukas@abaconcy:~$ netplan status -a
Online state: online
DNS Addresses: 127.0.0.53 (stub)
DNS Search: fritz.box

1: lo ethernet UNKNOWN/UP (unmanaged)
MAC Address: 00:00:00:00:00:00
Addresses: 127.0.0.1/8
::1/128
Routes: ::1 metric 256

6: sit0 tunnel/sit DOWN (unmanaged)

1: lan0 ethernet UP (networkd: usbC)
MAC Address: f8:e4:3b:2d:3b:b7 (ASIX Electronics Corp.)
Addresses: 192.168.178.117/24 (dhcp)
2001:9e8:alb0:f000:fae4:3bff:fe2d:3bb7/64
fe80::fae4:3bff:fe2d:3bb7/64 (link)
NS Addresses: 192.168.178.1
fd00::cece:leff:fe3d:c737
DNS Search: fritz.box
Routes: default via 192.168.178.1 from 192.168.178.117
192.168.178.0/24 from 192.168.178.117 metric 1
192.168.178.1 from 192.168.178.117 metric 1
2001:9e8:alb0:f000::/64 metric 100 (ra)
2001:9e8:alb0:f000::/56 via fe80::cece:leff:fe3d:c737 metric 100 (ra)
2001:dead:beef::/64 metric 100 (ra)
2a03:2260:300c:100::/64 metric 100 (ra)
fe80::/64 metric 256
default via fe80::cece:leff:fe3d:c737 metric 256

fbr0 bridge UP (unmanaged)
Address: 00:16:3e:0f:ee:29

hdb41f0e2 ethernet UP (unmanaged)
Address: b2:2d:a7:9a:bd:88

tunnel/sit UNKNOWN/UP (networkd: tun0)
Addresses: 2001:dead:beef::2/64
Routes: 2001:dead:beef::/64 metric 256

tunnel/wireguard UNKNOWN/UP (networkd: wg0)
Addresses: 10.10.0.2/24
Routes: 10.10.0.0/24 from 10.10.0.2 (link)
```

Section #2.1

# Status quo

# Status quo: Features

## Supported technologies

- Ethernet, InfiniBand
- WiFi + AP, cellular
- Bridge, Bond
- VLAN, veth, dummy
- Routing, PBR, VRF
- OVS, SR-IOV
- Tunnels (WireGuard, IPsec, VXLAN, GRE, ...)
- DNS, regulatory-domain

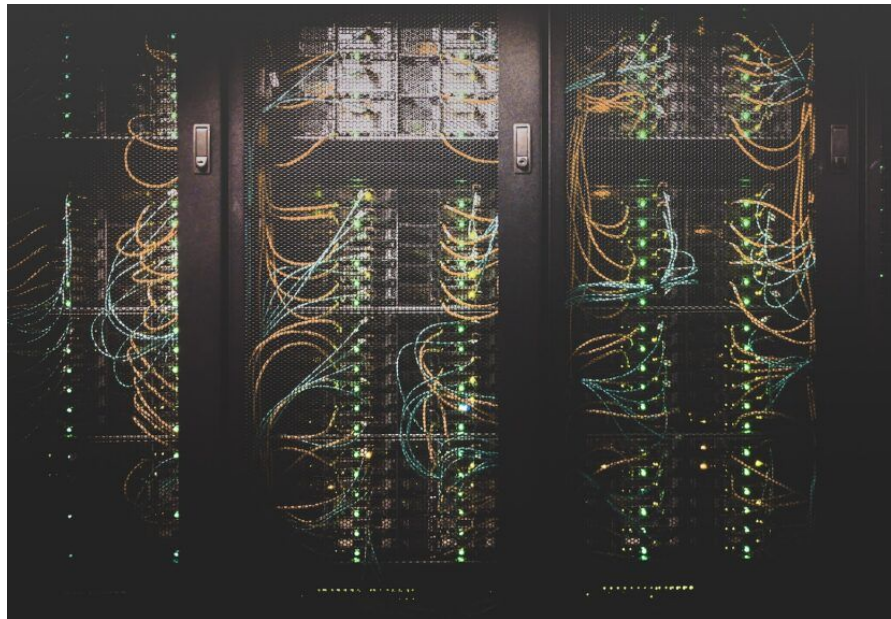


Photo by Taylor Vick ([Unsplash](#))

# Status quo: Extensions & Side-by-Side

## Native extensions

### **Systemd-networkd (override)**

`/run/systemd/network/10-netplan-eth0.network`

`/etc/systemd/network/10-netplan-eth0.network.d/override.conf`

### **NetworkManager (keyfile)**

YAML setting: `networkmanager.passthrough`

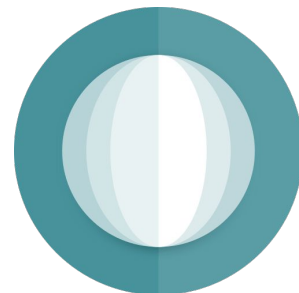
<https://netplan.io/reference#properties-for-device-type-nm-devices>

## Plays nice with 3rd parties

- Custom scripts
- iproute2
- Open vSwitch
- ifupdown{2,-ng}

# Status quo: d-i & cloud-images

- Netplan is used in Debian cloud-images since Bookworm (Kudos!)
  - Cloud-init integration
  - Playground for hands-on experience
    - [Netplan and systemd-networkd on Debian Bookworm – SlyBlog](#)
- ifupdown & NetworkManager in debian-installer
  - Many using systemd-networkd
  - Plenty of discussions: ifupdown{2,-ng} vs NetworkManager vs systemd-networkd
    - [DebConf'21 Cloud BoF](#)
    - [MiniDebConf'23 Hamburg](#)
    - [Re: Bug#995189: RFH: isc-dhcp](#)
    - [Re: proposal: dhcpcd-base as standard DHCP client starting with Trixie](#)

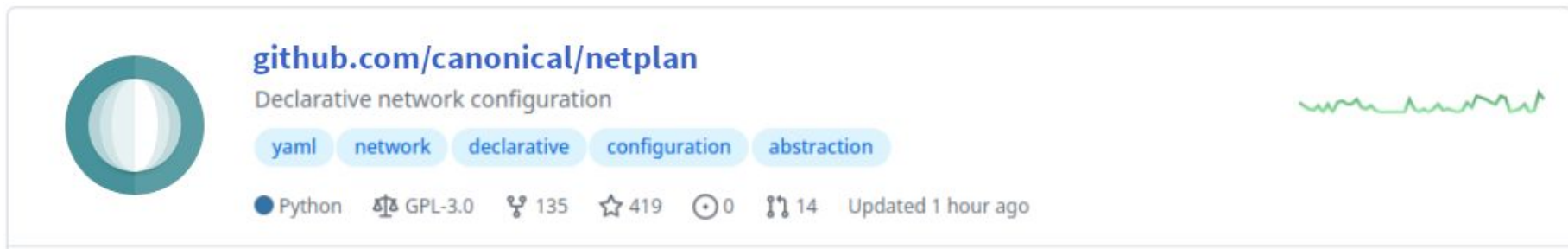


Section #3

# Future work

# Future work

- Netplan 1.0 release due 2024
  - stable API & ABI for 3rd party integrations
- Documentation improvements
  - [Salsa: debian/debian-reference \(!14\)](#)
- Debian-installer changes?
  - [Salsa: installer-team/netcfg \(!9\)](#)



The image shows the GitHub repository header for `canonical/netplan`. On the left is the repository's logo, a teal circle with a white vertical stripe. To its right is the repository name `github.com/canonical/netplan` in blue, followed by the description "Declarative network configuration". Below the description are five light blue tags: `yaml`, `network`, `declarative`, `configuration`, and `abstraction`. At the bottom, there is a row of icons and text: a Python logo, a license icon for GPL-3.0, a fork icon with the number 135, a star icon with the number 419, a clock icon with the number 0, a pull request icon with the number 14, and the text "Updated 1 hour ago". On the far right, there is a green line graph showing commit activity over time.

**github.com/canonical/netplan**  
Declarative network configuration

`yaml` `network` `declarative` `configuration` `abstraction`

Python GPL-3.0 135 419 0 14 Updated 1 hour ago



# Thank you. Questions?

Lukas Märdian – [slyon@debian.org](mailto:slyon@debian.org) | [slyon@ubuntu.com](mailto:slyon@ubuntu.com)

Ubuntu Foundations, Software Engineer @ Canonical



# FAQ

- Does it have a Python dependency?
  - No. Just the optional CLI, not the **netplan-generator** package.
- How are interactive changes in NetworkManager handled?
  - Bi-directional NetworkManager integration patch available.
- What about hooks?
  - Networkd-dispatcher, NM dispatcher (<https://netplan.io/faq#use-pre-up-post-up-etc-hook-scripts>)
- Is it just another competing standard?
  - No. It integrates nicely with native configuration & can be used in parallel.

# Netplan usage: CLI

```
Lukas@abaconcy:~$ netplan status
Online state: online
DNS Addresses: 127.0.0.53 (stub)
DNS Search: ff-sw.net

● 1: lo ethernet UNKNOWN/UP (unmanaged)
   MAC Address: 00:00:00:00:00:00
   Addresses: 127.0.0.1/8
              ::1/128
   Routes: ::1 metric 256

● 9: wlan0 wifi/"freifunk-weinstrasse.de" UP (NetworkManager: NM-b6b7a:
   MAC Address: f4:a4:75:a2:6c:5f (Intel Corporation)
   Addresses: 10.210.20.75/20
              2a03:2260:300c:100:3418:ec08:7f99:a525/64
              2a03:2260:300c:100:bfc5:b4af:a09e:a44b/64
              fe80::1b29:d4a5:c7a0:22ed/64 (link)
   DNS Addresses: 10.210.16.4
                  2a03:2260:300c:100::4
                  2a03:2260:300c:100::5
                  2a03:2260:300c:100::6
   DNS Search: ff-sw.net
   Routes: default via 10.210.16.4 metric 600 (dhcp)
            10.210.16.0/20 from 10.210.20.75 metric 600 (link)
            2a03:2260:300c:100::/64 metric 600 (ra)
            fe80::/64 metric 1024

7 inactive interfaces hidden. Use "--all" to show all.
```