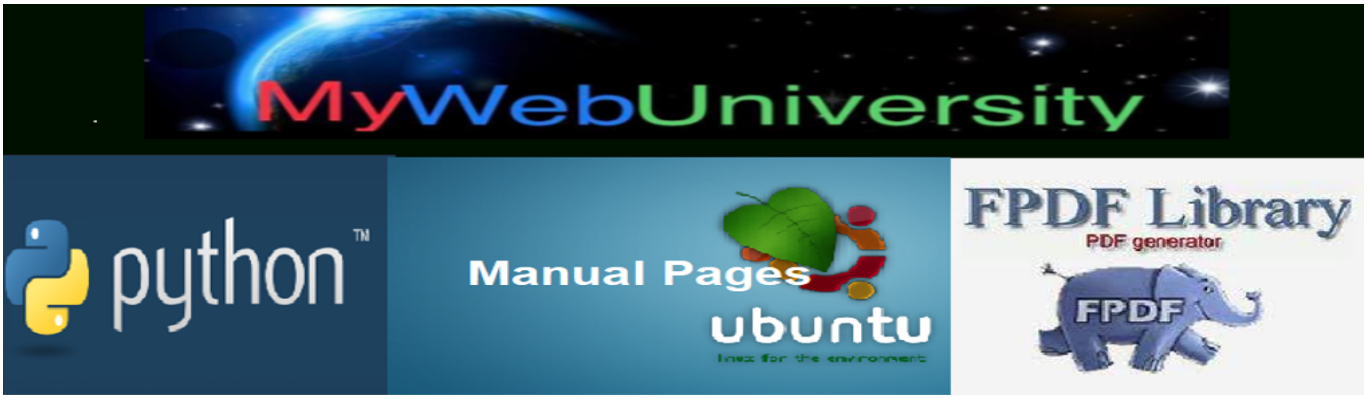

Rocky Enterprise Linux 9.2 Manual Pages on command 'netplan.5'

$ man netplan.5

YAML configuration(5) YAML configuration(5)

NAME

netplan - YAML network configuration abstraction for various backends

SYNOPSIS

netplan [COMMAND | help]

COMMANDS

See netplan help for a list of available commands on this system.

DESCRIPTION

Introduction

Distribution installers, cloud instantiation, image builds for particular devices, or any other way to deploy an operating system put its desired network configuration into YAML configuration file(s). During early boot, the netplan "network renderer" runs which reads /{lib,etc,run}/netplan/*.yaml and writes configuration to /run to hand off control of de? vices to the specified networking daemon.

? Configured devices get handled by systemd-networkd by default, unless explicitly marked as managed by a specific renderer (NetworkManager)

? Devices not covered by the network config do not get touched at all.

? Usable in initramfs (few dependencies and fast)

? No persistent generated config, only original YAML config

? Parser supports multiple config files to allow applications like libvirt or lxd to pack? age up expected network config (virbr0, lxdbr0), or to change the global default policy to use NetworkManager for everything.

? Retains the flexibility to change backends/policy later or adjust to removing Network?

Manager, as generated configuration is ephemeral.

General structure

netplan's configuration files use the YAML (http://yaml.org/spec/1.1/current.html) format. All /{lib,etc,run}/netplan/*.yaml are considered. Lexicographically later files (regard? less of in which directory they are) amend (new mapping keys) or override (same mapping keys) previous ones. A file in /run/netplan completely shadows a file with same name in /etc/netplan, and a file in either of those directories shadows a file with the same name in /lib/netplan.

The top-level node in a netplan configuration file is a network: mapping that contains version: 2 (the YAML currently being used by curtin, MaaS, etc. is version 1), and then device definitions grouped by their type, such as ethernets:, modems:, wifis:, or bridges:. These are the types that our renderer can understand and are supported by our backends.

Each type block contains device definitions as a map where the keys (called "configuration IDs") are defined as below.

Device configuration IDs

The key names below the per-device-type definition maps (like ethernets:) are called "ID"s. They must be unique throughout the entire set of configuration files. Their pri? mary purpose is to serve as anchor names for composite devices, for example to enumerate the members of a bridge that is currently being defined.

(Since 0.97) If an interface is defined with an ID in a configuration file; it will be brought up by the applicable renderer. To not have netplan touch an interface at all, it should be completely omitted from the netplan configuration files.

There are two physically/structurally different classes of device definitions, and the ID field has a different interpretation for each:

Physical devices

(Examples: ethernet, modem, wifi) These can dynamically come and go between reboots and even during runtime (hot plugging). In the generic case, they can be selected by match: rules on desired properties, such as name/name pattern, MAC address, driver, or device paths. In general these will match any number of devices (unless they refer to properties which are unique such as the full path or MAC address), so without further knowledge about the hardware these will always be considered as a group.

It is valid to specify no match rules at all, in which case the ID field is simply the interface name to be matched. This is mostly useful if you want to keep simple cases simple, and it's how network device configuration has been done for a long time.

If there are match: rules, then the ID field is a purely opaque name which is only being used for references from definitions of compound devices in the config.

Virtual devices

(Examples: veth, bridge, bond, vrf) These are fully under the control of the config file(s) and the network stack. I. e. these devices are being created instead of matched. Thus match: and set-name: are not applicable for these, and the ID field is the name of the created virtual device.

Top-level configuration structure

The general structure of a Netplan YAML file is shown below.

```
network:
  version: NUMBER
  renderer: STRING
  bonds: MAPPING
  bridges: MAPPING
  ethernets: MAPPING
  modems: MAPPING
  tunnels: MAPPING
  vlans: MAPPING
  vrfs: MAPPING
  wifis: MAPPING
  nm-devices: MAPPING
```

? version (number)

Defines what version of the configuration format is used. The only value sup? ported is 2. Defaults to 2 if not defined.

? renderer (scalar)

Defines what network configuration tool will be used to set up your configura? tion. Valid values are networkd and NetworkManager. Defaults to networkd if not defined.

? bonds (mapping)

Creates and configures link aggregation (bonding) devices.

? bridges (mapping)

Creates and configures bridge devices.

? ethernets (mapping)

Configures physical Ethernet interfaces.

? modems (mapping)

Configures modems

? tunnels (mapping)

Creates and configures different types of virtual tunnels.

? vlans (mapping)

Creates and configures VLANs.

? vrfs (mapping)

Configures Virtual Routing and Forwarding (VRF) devices.

? wifis (mapping)

Configures physical Wifi interfaces as client, adhoc or access point.

? nm-devices (mapping)

nm-devices are used in situations where Netplan doesn't support the connection type. The raw configuration expected by NetworkManager can be defined and will be passed as is (passthrough) to the .nmconnection file. Users will not normally use this type of device.

All the properties for all the device types will be described in the next sections.

Properties for physical device types

These properties are used with physical devices such as Ethernet and Wifi network inter? faces.

Note: Some options will not work reliably for devices matched by name only and rendered by networkd, due to interactions with device renaming in udev. Match devices by MAC when setting options like: wakeonlan or *-offload.

? match (mapping)

This selects a subset of available physical devices by various hardware proper? ties. The following configuration will then apply to all matching devices, as soon as they appear. All specified properties must match.

? name (scalar)

Current interface name. Globs are supported, and the primary use case for

matching on names, as selecting one fixed name can be more easily achieved with having no match: at all and just using the ID (see above). (NetworkManager: as of v1.14.0)

? macaddress (scalar)

Device's 6-byte permanent MAC address in the form "XX:XX:XX:XX:XX:XX" or 20 bytes for InfiniBand devices (IPoIB). Globs are not allowed. This doesn't match virtual MAC addresses for veth, bridge, bond, vlan, ...

? driver (scalar or sequence of scalars) ? sequence since 0.104

Kernel driver name, corresponding to the DRIVER udev property. A sequence of globs is supported, any of which must match. Matching on driver is only sup? ported with networkd.

Examples:

? All cards on second PCI bus:

```
network:
  ethernets:
    myinterface:
      match:
        name: enp2*
```

? Fixed MAC address:

```
network:
  ethernets:
    interface0:
      match:
        macaddress: 11:22:33:AA:BB:FF
```

? First card of driver ixgbe:

```
network:
  ethernets:
    nic0:
      match:
        driver: ixgbe
        name: en*s0
```

? First card with a driver matching bcmgenet or smsc*:

```
network:
```

```
ethernets:
  nic0:
    match:
      driver: ["bcmgenet", "smsc*"]
      name: en*
```

? set-name (scalar)

When matching on unique properties such as path or MAC, or with additional as? sumptions such as "there will only ever be one wifi device", match rules can be written so that they only match one device. Then this property can be used to give that device a more specific/desirable/nicer name than the default from udev's ifnames. Any additional device that satisfies the match rules will then fail to get renamed and keep the original kernel name (and dmesg will show an er? ror).

? wakeonlan (bool)

Enable wake on LAN. Off by default.

? emit-lldp (bool) ? since 0.99

(networkd backend only) Whether to emit LLDP packets. Off by default.

? receive-checksum-offload (bool) ? since 0.104

(networkd backend only) If set to true (false), the hardware offload for check? summing of ingress network packets is enabled (disabled). When unset, the ker? nel's default will be used.

? transmit-checksum-offload (bool) ? since 0.104

(networkd backend only) If set to true (false), the hardware offload for check? summing of egress network packets is enabled (disabled). When unset, the ker? nel's default will be used.

? tcp-segmentation-offload (bool) ? since 0.104

(networkd backend only) If set to true (false), the TCP Segmentation Offload (TSO) is enabled (disabled). When unset, the kernel's default will be used.

? tcp6-segmentation-offload (bool) ? since 0.104

(networkd backend only) If set to true (false), the TCP6 Segmentation Offload (tx-tcp6-segmentation) is enabled (disabled). When unset, the kernel's default will be used.

? generic-segmentation-offload (bool) ? since 0.104

(networkd backend only) If set to true (false), the Generic Segmentation Offload (GSO) is enabled (disabled). When unset, the kernel's default will be used.

? generic-receive-offload (bool) ? since 0.104

(networkd backend only) If set to true (false), the Generic Receive Offload (GRO) is enabled (disabled). When unset, the kernel's default will be used.

? large-receive-offload (bool) ? since 0.104

(networkd backend only) If set to true (false), the Large Receive Offload (LRO) is enabled (disabled). When unset, the kernel's default will be used.

? openvswitch (mapping) ? since 0.100

This provides additional configuration for the openvswitch network device. If Open vSwitch is not available on the system, netplan treats the presence of open? vswitch configuration as an error.

Any supported network device that is declared with the openvswitch mapping (or any bond/bridge that includes an interface with an openvswitch configuration) will be created in openvswitch instead of the defined renderer. In the case of a vlan definition declared the same way, netplan will create a fake VLAN bridge in openvswitch with the requested vlan properties.

? external-ids (mapping) ? since 0.100

Passed-through directly to Open vSwitch

? other-config (mapping) ? since 0.100

Passed-through directly to Open vSwitch

? lacp (scalar) ? since 0.100

Valid for bond interfaces. Accepts active, passive or off (the default).

? fail-mode (scalar) ? since 0.100

Valid for bridge interfaces. Accepts secure or standalone (the default).

? mcast-snooping (bool) ? since 0.100

Valid for bridge interfaces. False by default.

? protocols (sequence of scalars) ? since 0.100

Valid for bridge interfaces or the network section. List of protocols to be used when negotiating a connection with the controller. Accepts OpenFlow10, OpenFlow11, OpenFlow12, OpenFlow13, OpenFlow14, and OpenFlow15.

? rstp (bool) ? since 0.100

Valid for bridge interfaces. False by default.

? controller (mapping) ? since 0.100

Valid for bridge interfaces. Specify an external OpenFlow controller.

? addresses (sequence of scalars)

Set the list of addresses to use for the controller targets. The syntax of these addresses is as defined in ovs-vsctl(8). Example: addresses: [tcp:127.0.0.1:6653, "ssl:[fe80::1234%eth0]:6653"]

? connection-mode (scalar)

Set the connection mode for the controller. Supported options are in-band and out-of-band. The default is in-band.

? ports (sequence of sequence of scalars) ? since 0.100

Open vSwitch patch ports. Each port is declared as a pair of names which can be referenced as interfaces in dependent virtual devices (bonds, bridges).

Example:

```
openvswitch:
  ports:
    - [patch0-1, patch1-0]
```

? ssl (mapping) ? since 0.100

Valid for global openvswitch settings. Options for configuring SSL server end? point for the switch.

? ca-cert (scalar)

Path to a file containing the CA certificate to be used.

? certificate (scalar)

Path to a file containing the server certificate.

? private-key (scalar)

Path to a file containing the private key for the server.

Properties for all device types

? renderer (scalar)

Use the given networking backend for this definition. Currently supported are networkd and NetworkManager. This property can be specified globally in net? work:, for a device type (in e. g. ethernets:) or for a particular device defi? nition. Default is networkd.

(Since 0.99) The renderer property has one additional acceptable value for vlan objects (i. e. defined in vlans:): sriov. If a vlan is defined with the sriov

renderer for an SR-IOV Virtual Function interface, this causes netplan to set up a hardware VLAN filter for it. There can be only one defined per VF.

? dhcp4 (bool)

Enable DHCP for IPv4. Off by default.

? dhcp6 (bool)

Enable DHCP for IPv6. Off by default. This covers both stateless DHCP - where the DHCP server supplies information like DNS nameservers but not the IP address - and stateful DHCP, where the server provides both the address and the other in? formation.

If you are in an IPv6-only environment with completely stateless auto-configura? tion (SLAAC with RDNSS), this option can be set to cause the interface to be brought up. (Setting accept-ra alone is not sufficient.) Auto-configuration will still honor the contents of the router advertisement and only use DHCP if re? quested in the RA.

Note that rdnssd(8) is required to use RDNSS with networkd. No extra software is required for NetworkManager.

? ipv6-mtu (scalar) ? since 0.98 > Set the IPv6 MTU (only supported with networkd back? end). Note > that needing to set this is an unusual requirement. > > Requires feature: ipv6-mtu

? ipv6-privacy (bool)

Enable IPv6 Privacy Extensions (RFC 4941) for the specified interface, and prefer temporary addresses. Defaults to false - no privacy extensions. There is cur? rently no way to have a private address but prefer the public address.

? link-local (sequence of scalars)

Configure the link-local addresses to bring up. Valid options are 'ipv4' and 'ipv6', which respectively allow enabling IPv4 and IPv6 link local addressing.

If this field is not defined, the default is to enable only IPv6 link-local ad? dresses. If the field is defined but configured as an empty set, IPv6 link-local addresses are disabled as well as IPv4 link- local addresses.

This feature enables or disables link-local addresses for a protocol, but the ac? tual implementation differs per backend. On networkd, this directly changes the behavior and may add an extra address on an interface. When using the Network? Manager backend, enabling link-local has no effect if the interface also has DHCP

enabled.

Examples:

? Enable only IPv4 link-local: link-local: [ipv4]

? Enable all link-local addresses: link-local: [ipv4, ipv6]

? Disable all link-local addresses: link-local: [ ]

? ignore-carrier (bool) ? since 0.104

(networkd backend only) Allow the specified interface to be configured even if it has no carrier.

? critical (bool)

Designate the connection as "critical to the system", meaning that special care will be taken by to not release the assigned IP when the daemon is restarted. (not recognized by NetworkManager)

? dhcp-identifier (scalar)

(networkd backend only) Sets the source of DHCPv4 client identifier. If mac is specified, the MAC address of the link is used. If this option is omitted, or if duid is specified, networkd will generate an RFC4361-compliant client identifier for the interface by combining the link's IAID and DUID.

? dhcp4-overrides (mapping)

(networkd backend only) Overrides default DHCP behavior; see the DHCP Overrides section below.

? dhcp6-overrides (mapping)

(networkd backend only) Overrides default DHCP behavior; see the DHCP Overrides section below.

? accept-ra (bool)

Accept Router Advertisement that would have the kernel configure IPv6 by itself. When enabled, accept Router Advertisements. When disabled, do not respond to Router Advertisements. If unset use the host kernel default setting.

? addresses (sequence of scalars and mappings)

Add static addresses to the interface in addition to the ones received through DHCP or RA. Each sequence entry is in CIDR notation, i. e. of the form ad? dr/prefixlen. addr is an IPv4 or IPv6 address as recognized by inet_pton(3) and prefixlen the number of bits of the subnet.

For virtual devices (bridges, bonds, vlan) if there is no address configured and

DHCP is disabled, the interface may still be brought online, but will not be ad? dressable from the network.

In addition to the addresses themselves one can specify configuration parameters as mappings. Current supported options are:

? lifetime (scalar) ? since 0.100

Default: forever. This can be forever or 0 and corresponds to the Pre? ferredLifetime option in systemd-networkd's Address section. Currently sup? ported on the networkd backend only.

? label (scalar) ? since 0.100

An IP address label, equivalent to the ip address label command. Currently supported on the networkd backend only.

Examples:

? Simple: addresses: [192.168.14.2/24, "2001:1::1/64"]

? Advanced:

```
network:
  ethernets:
    eth0:
      addresses:
        - "10.0.0.15/24":
            lifetime: 0
            label: "maas"
        - "2001:1::1/64"
```

? ipv6-address-generation (scalar) ? since 0.99

Configure method for creating the address for use with RFC4862 IPv6 Stateless Ad? dress Auto-configuration (only supported with NetworkManager backend). Possible values are eui64 or stable-privacy.

? ipv6-address-token (scalar) ? since 0.100

Define an IPv6 address token for creating a static interface identifier for IPv6 Stateless Address Auto-configuration. This is mutually exclusive with ipv6-ad? dress-generation.

? gateway4, gateway6 (scalar)

Deprecated, see Default routes. Set default gateway for IPv4/6, for manual ad? dress configuration. This requires setting addresses too. Gateway IPs must be

in a form recognized by inet_pton(3). There should only be a single gateway per IP address family set in your global config, to make it unambiguous. If you need multiple default routes, please define them via routing-policy.

Examples

? IPv4: gateway4: 172.16.0.1

? IPv6: gateway6: "2001:4::1"

? nameservers (mapping)

Set DNS servers and search domains, for manual address configuration. There are two supported fields: addresses: is a list of IPv4 or IPv6 addresses similar to gateway*, and search: is a list of search domains.

Example:

```
network:
  ethernets:
    id0:
      [...]
      nameservers:
        search: [lab, home]
        addresses: [8.8.8.8, "FEDC::1"]
```

? macaddress (scalar)

Set the device's MAC address. The MAC address must be in the form "XX:XX:XX:XX:XX:XX".

Note: This will not work reliably for devices matched by name only and rendered by networkd, due to interactions with device renaming in udev. Match devices by MAC when setting MAC addresses.

Example:

```
network:
  ethernets:
    id0:
      match:
        macaddress: 52:54:00:6b:3c:58
      [...]
      macaddress: 52:54:00:6b:3c:59
```

? mtu (scalar)

Set the Maximum Transmission Unit for the interface. The default is 1500. Valid values depend on your network interface.

Note: This will not work reliably for devices matched by name only and rendered by networkd, due to interactions with device renaming in udev. Match devices by MAC when setting MTU.

? optional (bool)

An optional device is not required for booting. Normally, networkd will wait some time for device to become configured before proceeding with booting. However, if a device is marked as optional, networkd will not wait for it. This is only supported by networkd, and the default is false.

Example:

```
network:
  ethernets:
    eth7:
      # this is plugged into a test network that is often
      # down - don't wait for it to come up during boot.
      dhcp4: true
      optional: true
```

? optional-addresses (sequence of scalars)

Specify types of addresses that are not required for a device to be considered online. This changes the behavior of backends at boot time to avoid waiting for addresses that are marked optional, and thus consider the interface as "usable" sooner. This does not disable these addresses, which will be brought up anyway.

Example:

```
network:
  ethernets:
    eth7:
      dhcp4: true
      dhcp6: true
      optional-addresses: [ ipv4-ll, dhcp6 ]
```

? activation-mode (scalar) ? since 0.103

Allows specifying the management policy of the selected interface. By default, netplan brings up any configured interface if possible. Using the activation-

mode setting users can override that behavior by either specifying manual, to hand over control over the interface state to the administrator or (for networkd backend only) off to force the link in a down state at all times. Any interface with activation-mode defined is implicitly considered optional. Supported offi? cially as of networkd v248+.

Example:

```
network:
  ethernets:
    eth1:
      # this interface will not be put into an UP state automatically
      dhcp4: true
      activation-mode: manual
```

? routes (sequence of mappings)

Configure static routing for the device; see the Routing section below.

? routing-policy (sequence of mappings)

Configure policy routing for the device; see the Routing section below.

? neigh-suppress (scalar) ? since 0.105

Takes a boolean. Configures whether ARP and ND neighbor suppression is enabled for this port. When unset, the kernel's default will be used.

DHCP Overrides

Several DHCP behavior overrides are available. Most currently only have any effect when using the networkd backend, with the exception of use-routes and route-metric.

Overrides only have an effect if the corresponding dhcp4 or dhcp6 is set to true.

If both dhcp4 and dhcp6 are true, the networkd backend requires that dhcp4-overrides and dhcp6-overrides contain the same keys and values. If the values do not match, an error will be shown and the network configuration will not be applied.

When using the NetworkManager backend, different values may be specified for dhcp4-over? rides and dhcp6-overrides, and will be applied to the DHCP client processes as specified in the netplan YAML.

? dhcp4-overrides, dhcp6-overrides (mapping)

The dhcp4-overrides and `dhcp6-override`` mappings override the default DHCP be? havior.

? use-dns (bool)

Default: true. When true, the DNS servers received from the DHCP server will be used and take precedence over any statically configured ones. Currently only has an effect on the networkd backend.

? use-ntp (bool)

Default: true. When true, the NTP servers received from the DHCP server will be used by systemd-timesyncd and take precedence over any statically configured ones. Currently only has an effect on the networkd backend.

? send-hostname (bool)

Default: true. When true, the machine's hostname will be sent to the DHCP server. Currently only has an effect on the networkd backend.

? use-hostname (bool)

Default: true. When true, the hostname received from the DHCP server will be set as the transient hostname of the system. Currently only has an effect on the networkd backend.

? use-mtu (bool)

Default: true. When true, the MTU received from the DHCP server will be set as the MTU of the network interface. When false, the MTU advertised by the DHCP server will be ignored. Currently only has an effect on the networkd backend.

? hostname (scalar)

Use this value for the hostname which is sent to the DHCP server, instead of machine's hostname. Currently only has an effect on the networkd backend.

? use-routes (bool)

Default: true. When true, the routes received from the DHCP server will be installed in the routing table normally. When set to false, routes from the DHCP server will be ignored: in this case, the user is responsible for adding static routes if necessary for correct network operation. This allows users to avoid installing a default gateway for interfaces configured via DHCP. Available for both the networkd and NetworkManager backends.

? route-metric (scalar)

Use this value for default metric for automatically-added routes. Use this to prioritize routes for devices by setting a lower metric on a preferred interface. Available for both the networkd and NetworkManager backends.

? use-domains (scalar) ? since 0.98

Takes a boolean, or the special value "route". When true, the domain name re? ceived from the DHCP server will be used as DNS search domain over this link, similar to the effect of the Domains= setting. If set to "route", the domain name received from the DHCP server will be used for routing DNS queries only, but not for searching, similar to the effect of the Domains= setting when the argument is prefixed with "~".

Requires feature: dhcp-use-domains

Routing

Complex routing is possible with netplan. Standard static routes as well as policy rout? ing using routing tables are supported via the networkd backend.

These options are available for all types of interfaces.

Default routes

The most common need for routing concerns the definition of default routes to reach the wider Internet. Those default routes can only defined once per IP family and routing ta? ble. A typical example would look like the following:

```
network:
  ethernets:
    eth0:
      [...]
      routes:
        - to: default # could be 0.0.0.0/0 optionally
          via: 10.0.0.1
          metric: 100
          on-link: true
        - to: default # could be ::/0 optionally
          via: cf02:de:ad:be:ef::2
    eth1:
      [...]
      routes:
        - to: default
          via: 172.134.67.1
          metric: 100
          on-link: true
```

Not on the main routing table,

does not conflict with the eth0 default route

table: 76

? routes (mapping)

The routes block defines standard static routes for an interface. At least to must be specified. If type is local or nat a default scope of host is assumed. If type is unicast and no gateway (via) is given or type is broadcast, multicast or anycast a default scope of link is assumed. Otherwise, a global scope is the default setting.

For from, to, and via, both IPv4 and IPv6 addresses are recognized, and must be in the form addr/prefixlen or addr.

? from (scalar)

Set a source IP address for traffic going through the route. (NetworkManager: as of v1.8.0)

? to (scalar)

Destination address for the route.

? via (scalar)

Address to the gateway to use for this route.

? on-link (bool)

When set to "true", specifies that the route is directly connected to the in? terface. (NetworkManager: as of v1.12.0 for IPv4 and v1.18.0 for IPv6)

? metric (scalar)

The relative priority of the route. Must be a positive integer value.

? type (scalar)

The type of route. Valid options are "unicast" (default), "anycast", "black? hole", "broadcast", "local", "multicast", "nat", "prohibit", "throw", "unreach? able" or "xresolve".

? scope (scalar)

The route scope, how wide-ranging it is to the network. Possible values are "global", "link", or "host". Applies to IPv4 only.

? table (scalar)

The table number to use for the route. In some scenarios, it may be useful to set routes in a separate routing table. It may also be used to refer to rout?

ing policy rules which also accept a table parameter. Allowed values are posi?
tive integers starting from 1. Some values are already in use to refer to spe?
cific routing tables: see /etc/iproute2/rt_tables. (NetworkManager: as of
v1.10.0)

? mtu (scalar) ? since 0.101

The MTU to be used for the route, in bytes. Must be a positive integer value.

? congestion-window (scalar) ? since 0.102

The congestion window to be used for the route, represented by number of seg?
ments. Must be a positive integer value.

? advertised-receive-window (scalar) ? since 0.102

The receive window to be advertised for the route, represented by number of
segments. Must be a positive integer value.

? routing-policy (mapping)

The routing-policy block defines extra routing policy for a network, where traf?
fic may be handled specially based on the source IP, firewall marking, etc.
For from, to, both IPv4 and IPv6 addresses are recognized, and must be in the
form addr/prefixlen or addr.

? from (scalar)

Set a source IP address to match traffic for this policy rule.

? to (scalar)

Match on traffic going to the specified destination.

? table (scalar)

The table number to match for the route. In some scenarios, it may be useful
to set routes in a separate routing table. It may also be used to refer to
routes which also accept a table parameter. Allowed values are positive inte?
gers starting from 1. Some values are already in use to refer to specific
routing tables: see /etc/iproute2/rt_tables.

? priority (scalar)

Specify a priority for the routing policy rule, to influence the order in which
routing rules are processed. A higher number means lower priority: rules are
processed in order by increasing priority number.

? mark (scalar)

Have this routing policy rule match on traffic that has been marked by the ipt?

ables firewall with this value. Allowed values are positive integers starting from 1.

? type-of-service (scalar)

Match this policy rule based on the type of service number applied to the traf? fic.

Authentication

Netplan supports advanced authentication settings for ethernet and wifi interfaces, as well as individual wifi networks, by means of the auth block.

? auth (mapping)

Specifies authentication settings for a device of type ethernets:, or an access-points: entry on a wifis: device.

The auth block supports the following properties:

? key-management (scalar)

The supported key management modes are none (no key management); psk (WPA with pre-shared key, common for home wifi); eap (WPA with EAP, common for enterprise wifi); sae (used by WPA3); and 802.1x (used primarily for wired Ethernet con? nections).

? password (scalar)

The password string for EAP, or the pre-shared key for WPA-PSK.

The following properties can be used if key-management is eap or 802.1x:

? method (scalar)

The EAP method to use. The supported EAP methods are tls (TLS), peap (Protect? ed EAP), and ttls (Tunneled TLS).

? identity (scalar)

The identity to use for EAP.

? anonymous-identity (scalar)

The identity to pass over the unencrypted channel if the chosen EAP method sup? ports passing a different tunnelled identity.

? ca-certificate (scalar)

Path to a file with one or more trusted certificate authority (CA) certifi? cates.

? client-certificate (scalar)

Path to a file containing the certificate to be used by the client during au?

thentication.

? client-key (scalar)

Path to a file containing the private key corresponding to client-certificate.

? client-key-password (scalar)

Password to use to decrypt the private key specified in client-key if it is en?

crypted.

? phase2-auth (scalar) ? since 0.99

Phase 2 authentication mechanism.

Properties for device type ethernets:

Status: Optional.

Purpose: Use the ethernets key to configure Ethernet interfaces.

Structure: The key consists of a mapping of Ethernet interface IDs. Each ethernet has a number of configuration options. You don't need to define each interface by their name inside the ethernets mapping. You can use any ID that describes the interface and match the actual network card using the match key. The general configuration structure for Eth? ernets is shown below.

```
network:
  ethernets:
    device-id:
      ...
```

device-id is the interface identifier. If you use the interface name as the ID, Netplan will match that interface.

Consider the example below. In this case, an interface called eth0 will be configured with DHCP.

```
network:
  ethernets:
    eth0:
      dhcp4: true
```

The device-id can be any descriptive name your find meaningful. Although, if it doesn't match a real interface name, you must use the property match to identify the device you want to configure.

The example below defines an Ethernet connection called isp-interface (supposedly an ex? ternal interface connected to the Internet Service Provider) and uses match to apply the

configuration to the physical device with MAC address aa:bb:cc:00:11:22.

```
network:
  ethernets:
    isp-interface:
      match:
        macaddress: aa:bb:cc:00:11:22
      dhcp4: true
```

Ethernet device definitions, beyond common ones described above, also support some addi?
tional properties that can be used for SR-IOV devices.

? link (scalar) ? since 0.99

(SR-IOV devices only) The link property declares the device as a Virtual Function of the selected Physical Function device, as identified by the given netplan id.

Example:

```
network:
  ethernets:
    enp1: {...}
    enp1s16f1:
      link: enp1
```

? virtual-function-count (scalar) ? since 0.99

(SR-IOV devices only) In certain special cases VFs might need to be configured outside of netplan. For such configurations virtual-function-count can be op? tionally used to set an explicit number of Virtual Functions for the given Physi? cal Function. If unset, the default is to create only as many VFs as are defined in the netplan configuration. This should be used for special cases only.

Requires feature: sriov

? embedded-switch-mode (scalar) ? since 0.104

(SR-IOV devices only) Change the operational mode of the embedded switch of a supported SmartNIC PCI device (e.g. Mellanox ConnectX-5). Possible values are switchdev or legacy, if unspecified the vendor's default configuration is used.

Requires feature: eswitch-mode

? delay-virtual-functions-rebind (bool) ? since 0.104

(SR-IOV devices only) Delay rebinding of SR-IOV virtual functions to its driver after changing the embedded-switch-mode setting to a later stage. Can be enabled

when bonding/VF LAG is in use. Defaults to false.

Requires feature: eswitch-mode

? infiniband-mode (scalar) ? since 0.105

(InfiniBand devices only) Change the operational mode of a IPoIB device. Possi?

ble values are datagram or connected. If unspecified the kernel's default con?

figuration is used.

Requires feature: infiniband

Properties for device type modems:

Status: Optional.

Purpose: Use the modems key to configure Modem interfaces. GSM/CDMA modem configuration is only supported for the NetworkManager backend. systemd-networkd does not support modems.

Structure: The key consists of a mapping of Modem IDs. Each modem has a number of config? uration options. The general configuration structure for Modems is shown below.

```
network:
  version: 2
  renderer: NetworkManager
  modems:
    cdc-wdm1:
      mtu: 1600
      apn: ISP.CINGULAR
      username: ISP@CINGULARGPRS.COM
      password: CINGULAR1
      number: "*99#"
      network-id: 24005
      device-id: da812de91eec16620b06cd0ca5cbc7ea25245222
      pin: 2345
      sim-id: 89148000000060671234
      sim-operator-id: 310260
```

Requires feature: modems

? apn (scalar) ? since 0.99

Set the carrier APN (Access Point Name). This can be omitted if auto-config is enabled.

? auto-config (bool) ? since 0.99

Specify whether to try and auto-configure the modem by doing a lookup of the car?

rier against the Mobile Broadband Provider database. This may not work for all

carriers.

? device-id (scalar) ? since 0.99

Specify the device ID (as given by the WWAN management service) of the modem to

match. This can be found using mmcli.

? network-id (scalar) ? since 0.99

Specify the Network ID (GSM LAI format). If this is specified, the device will

not roam networks.

? number (scalar) ? since 0.99

The number to dial to establish the connection to the mobile broadband network.

(Deprecated for GSM)

? password (scalar) ? since 0.99

Specify the password used to authenticate with the carrier network. This can be

omitted if auto-config is enabled.

? pin (scalar) ? since 0.99

Specify the SIM PIN to allow it to operate if a PIN is set.

? sim-id (scalar) ? since 0.99

Specify the SIM unique identifier (as given by the WWAN management service) which

this connection applies to. If given, the connection will apply to any device

also allowed by device-id which contains a SIM card matching the given identifi?

er.

? sim-operator-id (scalar) ? since 0.99

Specify the MCC/MNC string (such as "310260" or "21601") which identifies the

carrier that this connection should apply to. If given, the connection will ap?

ply to any device also allowed by device-id and sim-id which contains a SIM card

provisioned by the given operator.

? username (scalar) ? since 0.99

Specify the username used to authenticate with the carrier network. This can be

omitted if auto-config is enabled.

Properties for device type wifis:

Status: Optional.

Purpose: Use the wifis key to configure WiFi access points.

Structure: The key consists of a mapping of WiFi IDs. Each wifi has a number of configu?

ration options. The general configuration structure for WiFis is shown below.

```
network:
  version: 2
  wifis:
    wlp0s1:
      access-points:
        "network_ssid_name":
          password: "**********"
```

Note that systemd-networkd does not natively support wifi, so you need wpasupplicant in?

stalled if you let the networkd renderer handle wifi.

? access-points (mapping)

This provides pre-configured connections to NetworkManager. Note that users can of course select other access points/SSIDs. The keys of the mapping are the SSIDs, and the values are mappings with the following supported properties:

? password (scalar)

Enable WPA/WPA2 authentication and set the passphrase for it. If neither this nor an auth block are given, the network is assumed to be open. The setting

```
password: "S3kr1t"
```

is equivalent to

```
auth:
  key-management: psk
  password: "S3kr1t"
```

? mode (scalar)

Possible access point modes are infrastructure (the default), ap (create an ac? cess point to which other devices can connect), and adhoc (peer to peer net? works without a central access point). ap is only supported with NetworkManag? er.

? bssid (scalar) ? since 0.99

If specified, directs the device to only associate with the given access point.

? band (scalar) ? since 0.99

Possible bands are 5GHz (for 5GHz 802.11a) and 2.4GHz (for 2.4GHz 802.11), do

not restrict the 802.11 frequency band of the network if unset (the default).

? channel (scalar) ? since 0.99

Wireless channel to use for the Wi-Fi connection. Because channel numbers overlap between bands, this property takes effect only if the band property is also set.

? hidden (bool) ? since 0.100

Set to true to change the SSID scan technique for connecting to hidden WiFi networks. Note this may have slower performance compared to false (the de? fault) when connecting to publicly broadcast SSIDs.

? wakeonwlan (sequence of scalars) ? since 0.99

This enables WakeOnWLan on supported devices. Not all drivers support all op? tions. May be any combination of any, disconnect, magic_pkt, gtk_rekey_failure, eap_identity_req, four_way_handshake, rfkill_release or tcp (NetworkManager on? ly). Or the exclusive default flag (the default).

? regulatory-domain (scalar) ? since 0.105

This can be used to define the radio's regulatory domain, to make use of addi? tional WiFi channels outside the "world domain". Takes an ISO / IEC 3166 country code (like GB) or 00 to reset to the "world domain". See wireless-regdb (https://git.kernel.org/pub/scm/linux/kernel/git/sforshee/wireless-regdb.git/tree/db.txt) for available values.

Requires dependency: iw, if it is to be used outside the networkd (wpa_suppli? cant) backend.

Properties for device type bridges:

Status: Optional.

Purpose: Use the bridges key to create Bridge interfaces.

Structure: The key consists of a mapping of Bridge interface names. Each bridge has an optional list of interfaces that will be bridged together. The interfaces listed in the interfaces key (enp5s0 and enp5s1 below) must also be defined in your Netplan configura? tion. The general configuration structure for Bridges is shown below.

```
network:
  bridges:
    br0:
      interfaces:
```

```
        - enp5s0
        - enp5s1
      dhcp4: true
      ...
```

When applied, a virtual interface of type bridge called br0 will be created in the system.

The specific settings for bridges are defined below.

? interfaces (sequence of scalars)

All devices matching this ID list will be added to the bridge. This may be an empty list, in which case the bridge will be brought online with no member inter? faces.

Example:

```
network:
  ethernets:
    switchports:
      match: {name: "enp2*"}
  [...]
  bridges:
    br0:
      interfaces: [switchports]
```

? parameters (mapping)

Customization parameters for special bridging options. Time intervals may need to be expressed as a number of seconds or milliseconds: the default value type is specified below. If necessary, time intervals can be qualified using a time suf? fix (such as "s" for seconds, "ms" for milliseconds) to allow for more control over its behavior.

? ageing-time, aging-time (scalar)

Set the period of time to keep a MAC address in the forwarding database after a packet is received. This maps to the AgeingTimeSec= property when the networkd renderer is used. If no time suffix is specified, the value will be interpret? ed as seconds.

? priority (scalar)

Set the priority value for the bridge. This value should be a number between 0 and 65535. Lower values mean higher priority. The bridge with the higher pri?

ority will be elected as the root bridge.

? port-priority (mapping)

Set the port priority per interface. The priority value is a number between 0 and 63. This metric is used in the designated port and root port selection al? gorithms.

Example:

```
network:
  ethernets:
    eth0:
      dhcp4: false
    eth1:
      dhcp4: false
  bridges:
    br0:
      interfaces: [eth0, eth1]
      parameters:
        port-priority:
          eth0: 10
          eth1: 20
```

? forward-delay (scalar)

Specify the period of time the bridge will remain in Listening and Learning states before getting to the Forward? DelaySec= property for the networkd renderer. If no time suffix is specified, the value will be interpreted as seconds.

? hello-time (scalar)

Specify the interval between two hello packets being sent out from the root and designated bridges. Hello packets communicate information about the network topology. When the networkd renderer is used, this maps to the HelloTimeSec= property. If no time suffix is specified, the value will be interpreted as seconds.

? max-age (scalar)

Set the maximum age of a hello packet. If the last hello packet is older than that value, the bridge will attempt to become the root bridge. This maps to

the MaxAgeSec= property when the networkd renderer is used. If no time suffix is specified, the value will be interpreted as seconds.

? path-cost (mapping)

Set the per-interface cost of a path on the bridge. Faster interfaces should have a lower cost. This allows a finer control on the network topology so that the fastest paths are available whenever possible.

Example:

```
network:
  ethernets:
    eth0:
      dhcp4: false
    eth1:
      dhcp4: false
  bridges:
    br0:
      interfaces: [eth0, eth1]
      parameters:
        path-cost:
          eth0: 100
          eth1: 200
```

? stp (bool)

Define whether the bridge should use Spanning Tree Protocol. The default value is "true", which means that Spanning Tree should be used.

Properties for device type bonds:

Status: Optional.

Purpose: Use the bonds key to create Bond (Link Aggregation) interfaces.

Structure: The key consists of a mapping of Bond interface names. Each bond has an op? tional list of interfaces that will be part of the aggregation. The interfaces listed in the interfaces key must also be defined in your Netplan configuration. The general con? figuration structure for Bonds is shown below.

```
network:
  bonds:
    bond0:
```

```
interfaces:
  - enp5s0
  - enp5s1
  - enp5s2
mode: active-backup
...
```

When applied, a virtual interface of type bond called bond0 will be created in the system.

The specific settings for bonds are defined below.

? interfaces (sequence of scalars)

All devices matching this ID list will be added to the bond.

Example:

```
network:
  ethernets:
    switchports:
      match: {name: "enp2*"}
  [...]
  bonds:
    bond0:
      interfaces: [switchports]
```

? parameters (mapping)

Customization parameters for special bonding options. Time intervals may need to be expressed as a number of seconds or milliseconds: the default value type is specified below. If necessary, time intervals can be qualified using a time suf? fix (such as "s" for seconds, "ms" for milliseconds) to allow for more control over its behavior.

? mode (scalar)

Set the bonding mode used for the interfaces. The default is balance-rr (round robin). Possible values are balance-rr, active-backup, balance-xor, broadcast, 802.3ad, balance-tlb, and balance-alb. For Open vSwitch active-backup and the additional modes balance-tcp and balance-slb are supported.

? lacp-rate (scalar)

Set the rate at which LACPDUs are transmitted. This is only useful in 802.3ad mode. Possible values are slow (30 seconds, default), and fast (every second).

? mii-monitor-interval (scalar)

Specifies the interval for MII monitoring (verifying if an interface of the bond has carrier). The default is 0; which disables MII monitoring. This is equivalent to the MIIMonitorSec= field for the networkd backend. If no time suffix is specified, the value will be interpreted as milliseconds.

? min-links (scalar)

The minimum number of links up in a bond to consider the bond interface to be up.

? transmit-hash-policy (scalar)

Specifies the transmit hash policy for the selection of ports. This is only useful in balance-xor, 802.3ad and balance-tlb modes. Possible values are lay? er2, layer3+4, layer2+3, encap2+3, and encap3+4.

? ad-select (scalar)

Set the aggregation selection mode. Possible values are stable, bandwidth, and count. This option is only used in 802.3ad mode.

? all-members-active (bool) ? since 0.106

If the bond should drop duplicate frames received on inactive ports, set this option to false. If they should be delivered, set this option to true. The default value is false, and is the desirable behavior in most situations.

Alias: all-slaves-active

? arp-interval (scalar)

Set the interval value for how frequently ARP link monitoring should happen. The default value is 0, which disables ARP monitoring. For the networkd back? end, this maps to the ARPIntervalSec= property. If no time suffix is speci? fied, the value will be interpreted as milliseconds.

? arp-ip-targets (sequence of scalars)

IPs of other hosts on the link which should be sent ARP requests in order to validate that a port is up. This option is only used when arp-interval is set to a value other than 0. At least one IP address must be given for ARP link monitoring to function. Only IPv4 addresses are supported. You can specify up to 16 IP addresses. The default value is an empty list.

? arp-validate (scalar)

Configure how ARP replies are to be validated when using ARP link monitoring.

Possible values are none, active, backup, and all.

? arp-all-targets (scalar)

Specify whether to use any ARP IP target being up as sufficient for a port to be considered up; or if all the targets must be up. This is only used for ac? tive-backup mode when arp-validate is enabled. Possible values are any and all.

? up-delay (scalar)

Specify the delay before enabling a link once the link is physically up. The default value is 0. This maps to the UpDelaySec= property for the networkd renderer. This option is only valid for the miimon link monitor. If no time suffix is specified, the value will be interpreted as milliseconds.

? down-delay (scalar)

Specify the delay before disabling a link once the link has been lost. The de? fault value is 0. This maps to the DownDelaySec= property for the networkd renderer. This option is only valid for the miimon link monitor. If no time suffix is specified, the value will be interpreted as milliseconds.

? fail-over-mac-policy (scalar)

Set whether to set all ports to the same MAC address when adding them to the bond, or how else the system should handle MAC addresses. The possible values are none, active, and follow.

? gratuitous-arp (scalar)

Specify how many ARP packets to send after failover. Once a link is up on a new port, a notification is sent and possibly repeated if this value is set to a number greater than 1. The default value is 1 and valid values are between 1 and 255. This only affects active-backup mode.

For historical reasons, the misspelling gratuitious-arp is also accepted and has the same function.

? packets-per-member (scalar) ? since 0.106

In balance-rr mode, specifies the number of packets to transmit on a port be? fore switching to the next. When this value is set to 0, ports are chosen at random. Allowable values are between 0 and 65535. The default value is 1. This setting is only used in balance-rr mode.

Alias: packets-per-slave

? primary-reselect-policy (scalar)

Set the reselection policy for the primary port. On failure of the active port, the system will use this policy to decide how the new active port will be chosen and how recovery will be handled. The possible values are always, bet? ter, and failure.

? resend-igmp (scalar)

In modes balance-rr, active-backup, balance-tlb and balance-alb, a failover can switch IGMP traffic from one port to another.

This parameter specifies how many IGMP membership reports are issued on a failover event. Values range from 0 to 255. 0 disables sending membership re? ports. Otherwise, the first membership report is sent on failover and subse? quent reports are sent at 200ms intervals.

? learn-packet-interval (scalar)

Specify the interval between sending learning packets to each port. The value range is between 1 and 0x7fffffff. The default value is 1. This option only affects balance-tlb and balance-alb modes. Using the networkd renderer, this field maps to the LearnPacketIntervalSec= property. If no time suffix is spec? ified, the value will be interpreted as seconds.

? primary (scalar)

Specify a device to be used as a primary port, or preferred device to use as a port for the bond (i.e. the preferred device to send data through), whenever it is available. This only affects active-backup, balance-alb, and balance-tlb modes.

Properties for device type tunnels:

Status: Optional.

Purpose: Use the tunnels key to create virtual tunnel interfaces.

Structure: The key consists of a mapping of tunnel interface names. Each tunnel requires the identification of the tunnel mode (see the section mode below for the list of support? ed modes). The general configuration structure for Tunnels is shown below.

```
network:
  tunnels:
    tunnel0:
      mode: SCALAR
```

...

When applied, a virtual interface called tunnel0 will be created in the system. Its oper? ation mode is defined by the property mode.

Tunnels allow traffic to pass as if it was between systems on the same local network, al? though systems may be far from each other but reachable via the Internet. They may be used to support IPv6 traffic on a network where the ISP does not provide the service, or to extend and "connect" separate local networks. Please see <https://en.wikipedia.org/wi? ki/Tunneling_protocol> for more general information about tunnels.

The specific settings for tunnels are defined below.

? mode (scalar)

Defines the tunnel mode. Valid options are sit, gre, ip6gre, ipip, ipip6, ip6ip6, vti, vti6, wireguard, vxlan, gretap and ip6gretap modes. In addition, the NetworkManager backend supports isatap tunnels.

? local (scalar)

Defines the address of the local endpoint of the tunnel. (For VXLAN) This should match one of the parent's IP addresses or make use of the networkd special val? ues.

? remote (scalar)

Defines the address of the remote endpoint of the tunnel or multicast group IP address for VXLAN.

? ttl (scalar) ? since 0.103

Defines the Time To Live (TTL) of the tunnel. Takes a number in the range 1..255.

? key (scalar or mapping)

Define keys to use for the tunnel. The key can be a number or a dotted quad (an IPv4 address). For wireguard it can be a base64-encoded private key or (as of networkd v242+) an absolute path to a file, containing the private key (since 0.100). It is used for identification of IP transforms. This is only required for vti and vti6 when using the networkd backend.

This field may be used as a scalar (meaning that a single key is specified and to be used for input, output and private key), or as a mapping, where you can fur? ther specify input/output/private.

? input (scalar)

The input key for the tunnel

? output (scalar)

The output key for the tunnel

? private (scalar) ? since 0.100

A base64-encoded private key required for WireGuard tunnels. When the systemd-networkd backend (v242+) is used, this can also be an absolute path to a file containing the private key.

? keys (scalar or mapping)

Alternate name for the key field. See above.

Examples:

```
network:
  tunnels:
    tun0:
      mode: gre
      local: ...
      remote: ...
      keys:
        input: 1234
        output: 5678
network:
  tunnels:
    tun0:
      mode: vti6
      local: ...
      remote: ...
      key: 59568549
network:
  tunnels:
    wg0:
      mode: wireguard
      addresses: [...]
      peers:
        - keys:
```

```
            public: rlbInAj0qV69CysWPQY7KEBnKxpYCpaWqOs/dLevdWc=
            shared: /path/to/shared.key
          ...
        key: mNb7OIIXTdgW4khM7OFlzJ+UPs7lmcWHV7xjPgakMkQ=
    network:
      tunnels:
        wg0:
          mode: wireguard
          addresses: [...]
          peers:
            - keys:
                public: rlbInAj0qV69CysWPQY7KEBnKxpYCpaWqOs/dLevdWc=
              ...
          keys:
            private: /path/to/priv.key
```

WireGuard specific keys:

? mark (scalar) ? since 0.100

Firewall mark for outgoing WireGuard packets from this interface, optional.

? port (scalar) ? since 0.100

UDP port to listen at or auto. Optional, defaults to auto.

? peers (sequence of mappings) ? since 0.100

A list of peers, each having keys documented below.

Example:

```
    network:
      tunnels:
        wg0:
          mode: wireguard
          key: /path/to/private.key
          mark: 42
          port: 5182
          peers:
            - keys:
                public: rlbInAj0qV69CysWPQY7KEBnKxpYCpaWqOs/dLevdWc=
```

```
      allowed-ips: [0.0.0.0/0, "2001:fe:ad:de:ad:be:ef:1/24"]
      keepalive: 23
      endpoint: 1.2.3.4:5
    - keys:
        public: M9nt4YujIOmNrRmpIRTmYSfMdrpvE7u6WkG8FY8WjG4=
        shared: /some/shared.key
      allowed-ips: [10.10.10.20/24]
      keepalive: 22
      endpoint: 5.4.3.2:1
```

? endpoint (scalar) ? since 0.100

Remote endpoint IPv4/IPv6 address or a hostname, followed by a colon and a port number.

? allowed-ips (sequence of scalars) ? since 0.100

A list of IP (v4 or v6) addresses with CIDR masks from which this peer is al? lowed to send incoming traffic and to which outgoing traffic for this peer is directed. The catch-all 0.0.0.0/0 may be specified for matching all IPv4 ad? dresses, and ::/0 may be specified for matching all IPv6 addresses.

? keepalive (scalar) ? since 0.100

An interval in seconds, between 1 and 65535 inclusive, of how often to send an authenticated empty packet to the peer for the purpose of keeping a stateful firewall or NAT mapping valid persistently. Optional.

? keys (mapping) ? since 0.100

Define keys to use for the WireGuard peers.

This field can be used as a mapping, where you can further specify the public and shared keys.

? public (scalar) ? since 0.100

A base64-encoded public key, required for WireGuard peers.

? shared (scalar) ? since 0.100

A base64-encoded preshared key. Optional for WireGuard peers. When the sys? temd-networkd backend (v242+) is used, this can also be an absolute path to a file containing the preshared key.

VXLAN specific keys:

? id (scalar) ? since 0.105

The VXLAN Network Identifier (VNI or VXLAN Segment ID). Takes a number in the range 1..16777215.

? link (scalar) ? since 0.105

netplan ID of the parent device definition to which this VXLAN gets connected.

? type-of-service (scalar) ? since 0.105

The Type Of Service byte value for a vxlan interface.

? mac-learning (scalar) ? since 0.105

Takes a boolean. When true, enables dynamic MAC learning to discover remote MAC addresses.

? ageing, aging (scalar) ? since 0.105

The lifetime of Forwarding Database entry learned by the kernel, in seconds.

? limit (scalar) ? since 0.105

Configures maximum number of FDB entries.

? arp-proxy (scalar) ? since 0.105

Takes a boolean. When true, bridge-connected VXLAN tunnel endpoint answers ARP requests from the local bridge on behalf of remote Distributed Overlay Virtual Ethernet (DOVE) clients. Defaults to false.

? notifications (sequence of scalars) ? since 0.105

Takes the flags l2-miss and l3-miss to enable netlink LLADDR and/or netlink IP address miss notifications.

? short-circuit (scalar) ? since 0.105

Takes a boolean. When true, route short circuiting is turned on.

? checksums (sequence of scalars) ? since 0.105

Takes the flags udp, zero-udp6-tx, zero-udp6-rx, remote-tx and remote-rx to en? able transmitting UDP checksums in VXLAN/IPv4, send/receive zero checksums in VXLAN/IPv6 and enable sending/receiving checksum offloading in VXLAN.

? extensions (sequence of scalars) ? since 0.105

Takes the flags group-policy and generic-protocol to enable the "Group Policy" and/or "Generic Protocol" VXLAN extensions.

? port (scalar) ? since 0.105

Configures the default destination UDP port. If the destination port is not specified then Linux kernel default will be used. Set to 4789 to get the IANA assigned value.

? port-range (sequence of scalars) ? since 0.105

Configures the source port range for the VXLAN. The kernel assigns the source UDP port based on the flow to help the receiver to do load balancing. When this option is not set, the normal range of local UDP ports is used. Uses the form [LOWER, UPPER].

? flow-label (scalar) ? since 0.105

Specifies the flow label to use in outgoing packets. The valid range is 0-1048575.

? do-not-fragment (scalar) ? since 0.105

Allows setting the IPv4 Do not Fragment (DF) bit in outgoing packets. Takes a boolean value. When unset, the kernel's default will be used.

Properties for device type vlans:

Status: Optional.

Purpose: Use the vlans key to create VLAN interfaces.

Structure: The key consists of a mapping of VLAN interface names. The interface used in the link option (enp5s0 in the example below) must also be defined in the Netplan configu? ration. The general configuration structure for Vlans is shown below.

```
network:
  vlans:
    vlan123:
      id: 123
      link: enp5s0
      dhcp4: yes
```

The specific settings for VLANs are defined below.

? id (scalar)

VLAN ID, a number between 0 and 4094.

? link (scalar)

netplan ID of the underlying device definition on which this VLAN gets created.

Example:

```
network:
  ethernets:
    eno1: {...}
  vlans:
```

```
    en-intra:
      id: 1
      link: eno1
      dhcp4: yes
    en-vpn:
      id: 2
      link: eno1
      addresses: [...]
```

Properties for device type vrfs:

Status: Optional.

Purpose: Use the vrfs key to create Virtual Routing and Forwarding (VRF) interfaces.

Structure: The key consists of a mapping of VRF interface names. The interface used in the link option (enp5s0 in the example below) must also be defined in the Netplan configu? ration. The general configuration structure for VRFs is shown below.

```
  network:
    renderer: networkd
    vrfs:
      vrf1:
        table: 1
        interfaces:
          - enp5s0
        routes:
          - to: default
            via: 10.10.10.4
        routing-policy:
          - from: 10.10.10.42
```

? table (scalar) ? since 0.105

The numeric routing table identifier. This setting is compulsory.

? interfaces (sequence of scalars) ? since 0.105

All devices matching this ID list will be added to the VRF. This may be an empty list, in which case the VRF will be brought online with no member interfaces.

? routes (sequence of mappings) ? since 0.105

Configure static routing for the device; see the Routing section. The table val?

ue is implicitly set to the VRF's table.

? routing-policy (sequence of mappings) ? since 0.105

Configure policy routing for the device; see the Routing section. The table val?

ue is implicitly set to the VRF's table.

Example:

```
network:
  vrfs:
    vrf20:
      table: 20
      interfaces: [ br0 ]
      routes:
        - to: default
          via: 10.10.10.3
      routing-policy:
        - from: 10.10.10.42
    [...]
  bridges:
    br0:
      interfaces: []
```

Properties for device type nm-devices:

Status: Optional. Its use is not recommended.

Purpose: Use the nm-devices key to configure device types that are not supported by Net? plan. This is NetworkManager specific configuration.

Structure: The key consists of a mapping of NetworkManager connections. The nm-devices device type is for internal use only and should not be used in normal configuration files.

It enables a fallback mode for unsupported settings, using the passthrough mapping. The general configuration structure for NM connections is shown below.

```
network:
  version: 2
  nm-devices:
    NM-db5f0f67-1f4c-4d59-8ab8-3d278389cf87:
      renderer: NetworkManager
      networkmanager:
```

uuid: "db5f0f67-1f4c-4d59-8ab8-3d278389cf87"

name: "myvpnconnection"

passthrough:

connection.type: "vpn"

vpn.ca: "path to ca.crt"

vpn.cert: "path to client.crt"

vpn.cipher: "AES-256-GCM"

vpn.connection-type: "tls"

vpn.dev: "tun"

vpn.key: "path to client.key"

vpn.remote: "1.2.3.4:1194"

vpn.service-type: "org.freedesktop.NetworkManager.openvpn"

Backend-specific configuration parameters

In addition to the other fields available to configure interfaces, some backends may re? quire to record some of their own parameters in netplan, especially if the netplan defini? tions are generated automatically by the consumer of that backend. Currently, this is on? ly used with NetworkManager.

? networkmanager (mapping) ? since 0.99

Keeps the NetworkManager-specific configuration parameters used by the daemon to recognize connections.

? name (scalar) ? since 0.99

Set the display name for the connection.

? uuid (scalar) ? since 0.99

Defines the UUID (unique identifier) for this connection, as generated by Net? workManager itself.

? stable-id (scalar) ? since 0.99

Defines the stable ID (a different form of a connection name) used by Network? Manager in case the name of the connection might otherwise change, such as when sharing connections between users.

? device (scalar) ? since 0.99

Defines the interface name for which this connection applies.

? passthrough (mapping) ? since 0.102

Can be used as a fallback mechanism to missing keyfile settings.

SEE ALSO

netplan-generate(8), netplan-apply(8), netplan-try(8), netplan-get(8), netplan-set(8), netplan-info(8), netplan-ip(8), netplan-rebind(8), netplan-status(8), netplan-dbus(8), systemd-networkd(8), NetworkManager(8)

AUTHORS

Mathieu Trudel-Lapierre (<cyphermox@ubuntu.com>); Martin Pitt (<martin.pitt@ubuntu.com>); Lukas M?rdian (<slyon@ubuntu.com>).