[Skip to main content](https://netplan.io/examples/#main-content)

* [Products](https://netplan.io/examples/#canonical-products)

[Netplan](https://netplan.io/)

[Jump to main content](https://netplan.io/examples/#main-content)

* [Reference](https://netplan.io/reference)
* [Design](https://netplan.io/design)
* [Examples](https://netplan.io/examples)
* [FAQ](https://netplan.io/faq)
* [Troubleshooting](https://netplan.io/troubleshooting)

# Netplan configuration examples

Below are a collection of example netplan configurations for common scenarios. If you see a scenario missing or have one to contribute, please file a bug against this documentation with the example using the links at the bottom of this page. Thank you!

## Configuration

To configure netplan, save configuration files under /etc/netplan/ with a .yaml extension (e.g. /etc/netplan/config.yaml), then run sudo netplan apply. This command parses and applies the configuration to the system. Configuration written to disk under /etc/netplan/ will persist between reboots.

## Using DHCP and static addressing

To let the interface named ‘enp3s0’ get an address via DHCP, create a YAML file with the following:

network:

version: 2

renderer: networkd

ethernets:

enp3s0:

dhcp4: true

To instead set a static IP address, use the addresses key, which takes a list of (IPv4 or IPv6), addresses along with the subnet prefix length (e.g. /24). DNS information can be provided as well, and the gateway can be defined via a default route:

network:

version: 2

renderer: networkd

ethernets:

enp3s0:

addresses:

- 10.10.10.2/24

nameservers:

search: [mydomain, otherdomain]

addresses: [10.10.10.1, 1.1.1.1]

routes:

- to: default

via: 10.10.10.1

## Connecting multiple interfaces with DHCP

Many systems now include more than one network interface. Servers will commonly need to connect to multiple networks, and may require that traffic to the Internet goes through a specific interface despite all of them providing a valid gateway.

One can achieve the exact routing desired over DHCP by specifying a metric for the routes retrieved over DHCP, which will ensure some routes are preferred over others. In this example, ‘enred’ is preferred over ‘engreen’, as it has a lower route metric:

network:

version: 2

ethernets:

enred:

dhcp4: yes

dhcp4-overrides:

route-metric: 100

engreen:

dhcp4: yes

dhcp4-overrides:

route-metric: 200

## Connecting to an open wireless network

Netplan easily supports connecting to an open wireless network (one that is not secured by a password), only requiring that the access point is defined:

network:

version: 2

wifis:

wl0:

access-points:

opennetwork: {}

dhcp4: yes

## Connecting to a WPA Personal wireless network

Wireless devices use the ‘wifis’ key and share the same configuration options with wired ethernet devices. The wireless access point name and password should also be specified:

network:

version: 2

renderer: networkd

wifis:

wlp2s0b1:

dhcp4: no

dhcp6: no

addresses: [192.168.0.21/24]

nameservers:

addresses: [192.168.0.1, 8.8.8.8]

access-points:

"network\_ssid\_name":

password: "\*\*\*\*\*\*\*\*\*\*"

routes:

- to: default

via: 192.168.0.1

## Connecting to WPA Enterprise wireless networks

It is also common to find wireless networks secured using WPA or WPA2 Enterprise, which requires additional authentication parameters.

For example, if the network is secured using WPA-EAP and TTLS:

network:

version: 2

wifis:

wl0:

access-points:

workplace:

auth:

key-management: eap

method: ttls

anonymous-identity: "@internal.example.com"

identity: "joe@internal.example.com"

password: "v3ryS3kr1t"

dhcp4: yes

Or, if the network is secured using WPA-EAP and TLS:

network:

version: 2

wifis:

wl0:

access-points:

university:

auth:

key-management: eap

method: tls

anonymous-identity: "@cust.example.com"

identity: "cert-joe@cust.example.com"

ca-certificate: /etc/ssl/cust-cacrt.pem

client-certificate: /etc/ssl/cust-crt.pem

client-key: /etc/ssl/cust-key.pem

client-key-password: "d3cryptPr1v4t3K3y"

dhcp4: yes

Many different modes of encryption are supported. See the [Netplan reference](https://netplan.io/reference) page.

## Using multiple addresses on a single interface

The addresses key can take a list of addresses to assign to an interface:

network:

version: 2

renderer: networkd

ethernets:

enp3s0:

addresses:

- 10.100.1.38/24

- 10.100.1.39/24

routes:

- to: default

via: 10.100.1.1

Interface aliases (e.g. eth0:0) are not supported.

## Using multiple addresses with multiple gateways

Similar to the example above, interfaces with multiple addresses can be

configured with multiple gateways.

network:

version: 2

renderer: networkd

ethernets:

enp3s0:

addresses:

- 10.0.0.10/24

- 11.0.0.11/24

routes:

- to: default

via: 10.0.0.1

metric: 200

- to: default

via: 11.0.0.1

metric: 300

We configure individual routes to default (or 0.0.0.0/0) using the address of the gateway for the subnet. The metric value should be adjusted so the routing happens as expected.

DHCP can be used to receive one of the IP addresses for the interface. In this case, the default route for that address will be automatically configured with a metric value of 100.

## Using Network Manager as a renderer

Netplan supports both networkd and Network Manager as backends. You can specify which network backend should be used to configure particular devices by using the renderer key. You can also delegate all configuration of the network to Network Manager itself by specifying only the renderer key:

network:

version: 2

renderer: NetworkManager

## Configuring interface bonding

Bonding is configured by declaring a bond interface with a list of physical interfaces and a bonding mode. Below is an example of an active-backup bond that uses DHCP to obtain an address:

network:

version: 2

renderer: networkd

bonds:

bond0:

dhcp4: yes

interfaces:

- enp3s0

- enp4s0

parameters:

mode: active-backup

primary: enp3s0

Below is an example of a system acting as a router with various bonded interfaces and different types. Note the ‘optional: true’ key declarations that allow booting to occur without waiting for those interfaces to activate fully.

network:

version: 2

renderer: networkd

ethernets:

enp1s0:

dhcp4: no

enp2s0:

dhcp4: no

enp3s0:

dhcp4: no

optional: true

enp4s0:

dhcp4: no

optional: true

enp5s0:

dhcp4: no

optional: true

enp6s0:

dhcp4: no

optional: true

bonds:

bond-lan:

interfaces: [enp2s0, enp3s0]

addresses: [192.168.93.2/24]

parameters:

mode: 802.3ad

mii-monitor-interval: 1

bond-wan:

interfaces: [enp1s0, enp4s0]

addresses: [192.168.1.252/24]

nameservers:

search: [local]

addresses: [8.8.8.8, 8.8.4.4]

parameters:

mode: active-backup

mii-monitor-interval: 1

gratuitious-arp: 5

routes:

- to: default

via: 192.168.1.1

bond-conntrack:

interfaces: [enp5s0, enp6s0]

addresses: [192.168.254.2/24]

parameters:

mode: balance-rr

mii-monitor-interval: 1

## Configuring network bridges

To create a very simple bridge consisting of a single device that uses DHCP, write:

network:

version: 2

renderer: networkd

ethernets:

enp3s0:

dhcp4: no

bridges:

br0:

dhcp4: yes

interfaces:

- enp3s0

A more complex example, to get libvirtd to use a specific bridge with a tagged vlan, while continuing to provide an untagged interface as well would involve:

network:

version: 2

renderer: networkd

ethernets:

enp0s25:

dhcp4: true

bridges:

br0:

addresses: [ 10.3.99.25/24 ]

interfaces: [ vlan15 ]

vlans:

vlan15:

accept-ra: no

id: 15

link: enp0s25

Then libvirtd would be configured to use this bridge by adding the following content to a new XML file under /etc/libvirtd/qemu/networks/. The name of the bridge in the <bridge> tag as well as in <name> need to match the name of the bridge device configured using netplan:

<network>

<name>br0</name>

<bridge name='br0'/>

<forward mode="bridge"/>

</network>

## Attaching VLANs to network interfaces

To configure multiple VLANs with renamed interfaces:

network:

version: 2

renderer: networkd

ethernets:

mainif:

match:

macaddress: "de:ad:be:ef:ca:fe"

set-name: mainif

addresses: [ "10.3.0.5/23" ]

nameservers:

addresses: [ "8.8.8.8", "8.8.4.4" ]

search: [ example.com ]

routes:

- to: default

via: 10.3.0.1

vlans:

vlan15:

id: 15

link: mainif

addresses: [ "10.3.99.5/24" ]

vlan10:

id: 10

link: mainif

addresses: [ "10.3.98.5/24" ]

nameservers:

addresses: [ "127.0.0.1" ]

search: [ domain1.example.com, domain2.example.com ]

## Reaching a directly connected gateway

This allows setting up a default route, or any route, using the “on-link” keyword where the gateway is an IP address that is directly connected to the network even if the address does not match the subnet configured on the interface.

network:

version: 2

renderer: networkd

ethernets:

ens3:

addresses: [ "10.10.10.1/24" ]

routes:

- to: default # or 0.0.0.0/0

via: 9.9.9.9

on-link: true

For IPv6 the config would be very similar, with the notable difference being an additional scope: link host route to the router’s address required:

network:

version: 2

renderer: networkd

ethernets:

ens3:

addresses: [ "2001:cafe:face:beef::dead:dead/64" ]

routes:

- to: "2001:cafe:face::1/128"

scope: link

- to: default # or "::/0"

via: "2001:cafe:face::1"

on-link: true

## Configuring source routing

Route tables can be added to particular interfaces to allow routing between two networks:

In the example below, ens3 is on the 192.168.3.0/24 network and ens5 is on the 192.168.5.0/24 network. This enables clients on either network to connect to the other and allow the response to come from the correct interface.

Furthermore, the default route is still assigned to ens5 allowing any other traffic to go through it.

network:

version: 2

renderer: networkd

ethernets:

ens3:

addresses:

- 192.168.3.30/24

dhcp4: no

routes:

- to: 192.168.3.0/24

via: 192.168.3.1

table: 101

routing-policy:

- from: 192.168.3.0/24

table: 101

ens5:

addresses:

- 192.168.5.24/24

dhcp4: no

routes:

- to: default

via: 192.168.5.1

- to: 192.168.5.0/24

via: 192.168.5.1

table: 102

routing-policy:

- from: 192.168.5.0/24

table: 102

## Configuring a loopback interface

Networkd does not allow creating new loopback devices, but a user can add new addresses to the standard loopback interface, lo, in order to have it considered a valid address on the machine as well as for custom routing:

network:

version: 2

renderer: networkd

ethernets:

lo:

match:

name: lo

addresses: [ 7.7.7.7/32 ]

## Integration with a Windows DHCP Server

For networks where DHCP is provided by a Windows Server using the dhcp-identifier key allows for interoperability:

network:

version: 2

ethernets:

enp3s0:

dhcp4: yes

dhcp-identifier: mac

## Connecting an IP tunnel

Tunnels allow an administrator to extend networks across the Internet by configuring two endpoints that will connect a special tunnel interface and do the routing required. Netplan supports SIT, GRE, IP-in-IP (ipip, ipip6, ip6ip6), IP6GRE, VTI and VTI6 tunnels.

A common use of tunnels is to enable IPv6 connectivity on networks that only support IPv4. The example below show how such a tunnel might be configured.

Here, 1.1.1.1 is the client’s own IP address; 2.2.2.2 is the remote server’s IPv4 address, “2001:dead:beef::2/64” is the client’s IPv6 address as defined by the tunnel, and “2001:dead:beef::1” is the remote server’s IPv6 address.

Finally, “2001:cafe:face::1/64” is an address for the client within the routed IPv6 prefix:

network:

version: 2

ethernets:

eth0:

addresses:

- 1.1.1.1/24

- "2001:cafe:face::1/64"

routes:

- to: default

via: 1.1.1.254

tunnels:

he-ipv6:

mode: sit

remote: 2.2.2.2

local: 1.1.1.1

addresses:

- "2001:dead:beef::2/64"

routes:

- to: default

via: "2001:dead:beef::1"

## Configuring SR-IOV Virtual Functions

For SR-IOV network cards, it is possible to dynamically allocate Virtual Function interfaces for every configured Physical Function. In netplan, a VF is defined by having a link: property pointing to the parent PF.

network:

version: 2

ethernets:

eno1:

mtu: 9000

enp1s16f1:

link: eno1

addresses : [ "10.15.98.25/24" ]

vf1:

match:

name: enp1s16f[2-3]

link: eno1

addresses : [ "10.15.99.25/24" ]

### TABLE OF CONTENTS

* [Configuration](https://netplan.io/examples/#configuration)
* [Using DHCP and static addressing](https://netplan.io/examples/#using-dhcp-and-static-addressing)
* [Connecting multiple interfaces with DHCP](https://netplan.io/examples/#connecting-multiple-interfaces-with-dhcp)
* [Connecting to an open wireless network](https://netplan.io/examples/#connecting-to-an-open-wireless-network)
* [Connecting to a WPA Personal wireless network](https://netplan.io/examples/#connecting-to-a-wpa-personal-wireless-network)
* [Connecting to WPA Enterprise wireless networks](https://netplan.io/examples/#connecting-to-wpa-enterprise-wireless-networks)
* [Using multiple addresses on a single interface](https://netplan.io/examples/#using-multiple-addresses-on-a-single-interface)
* [Using multiple addresses with multiple gateways](https://netplan.io/examples/#using-multiple-addresses-with-multiple-gateways)
* [Using Network Manager as a renderer](https://netplan.io/examples/#using-network-manager-as-a-renderer)
* [Configuring interface bonding](https://netplan.io/examples/#configuring-interface-bonding)
* [Configuring network bridges](https://netplan.io/examples/#configuring-network-bridges)
* [Attaching VLANs to network interfaces](https://netplan.io/examples/#attaching-vlans-to-network-interfaces)
* [Reaching a directly connected gateway](https://netplan.io/examples/#reaching-a-directly-connected-gateway)
* [Configuring source routing](https://netplan.io/examples/#configuring-source-routing)
* [Configuring a loopback interface](https://netplan.io/examples/#configuring-a-loopback-interface)
* [Integration with a Windows DHCP Server](https://netplan.io/examples/#integration-with-a-windows-dhcp-server)
* [Connecting an IP tunnel](https://netplan.io/examples/#connecting-an-ip-tunnel)
* [Configuring SR-IOV Virtual Functions](https://netplan.io/examples/#configuring-sr-iov-virtual-functions)

#### Netplan

* [Home](https://netplan.io/)
* [Reference](https://netplan.io/reference)
* [Design](https://netplan.io/design)
* [Examples](https://netplan.io/examples)

#### Need help?

* [Ask Ubuntu](https://askubuntu.com/questions/tagged/netplan)
* [Join #netplan on Libera Chat](https://libera.chat/)
* [Report a bug](https://launchpad.net/netplan/+filebug)

#### Contribute

* [GitHub](https://github.com/canonical/netplan)

© 2022 Canonical Ltd. Ubuntu and Canonical are registered trademarks of Canonical Ltd.

* [Legal info](https://www.ubuntu.com/legal)
* [Manage your tracker settings](https://netplan.io/examples/)
* [Report a bug with this site](https://github.com/canonical-websites/netplan.io/issues/new)
* [Report a bug with Netplan](https://launchpad.net/netplan/+filebug)

[Go to the top of the page](https://netplan.io/examples/#)