**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.

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.

Also, see [/examples](https://github.com/canonical/netplan/tree/main/examples) on GitHub.

**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://people.ubuntu.com/~slyon/netplan-docs/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:**

**-** 10.100.1.37/24

**-** **10.100.1.38/24:**

**label:** "enp3s0:0"

**-** **10.100.1.39/24:**

**label:** "enp3s0:some-label"

**routes:**

**-** **to:** default

**via:** 10.100.1.1

**Using multiple addresses with multiple gateways**

Similar to the example above, interfaces with multiple addresses can be configured with multiple gateways, and static DNS nameservers (Google DNS for this example):

**network:**

**version:** 2

**renderer:** networkd

**ethernets:**

**enp3s0:**

**addresses:**

**-** 10.0.0.10/24

**-** 11.0.0.11/24

**nameservers:**

**addresses:**

**-** 8.8.8.8

**-** 8.8.4.4

**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:**

**addresses:** **[** "127.0.0.1/8"**,** "::1/128"**,** "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" **]**

**Complex example**

This is a complex example which shows most available features

**network:**

**version:** 2

*# if specified, can only realistically have that value, as networkd cannot*

*# render wifi/3G.*

**renderer:** NetworkManager

**ethernets:**

*# opaque ID for physical interfaces, only referred to by other stanzas*

**id0:**

**match:**

**macaddress:** 00:11:22:33:44:55

**wakeonlan:** true

**dhcp4:** true

**addresses:**

**-** 192.168.14.2/24

**-** 192.168.14.3/24

**-** "2001:1::1/64"

**nameservers:**

**search:** **[**foo.local**,** bar.local**]**

**addresses:** **[**8.8.8.8**]**

**routes:**

**-** **to:** default

**via:** 192.168.14.1

**-** **to:** default

**via:** "2001:1::2"

**-** **to:** 0.0.0.0/0

**via:** 11.0.0.1

**table:** 70

**on-link:** true

**metric:** 3

**routing-policy:**

**-** **to:** 10.0.0.0/8

**from:** 192.168.14.2/24

**table:** 70

**priority:** 100

**-** **to:** 20.0.0.0/8

**from:** 192.168.14.3/24

**table:** 70

**priority:** 50

*# only networkd can render on-link routes and routing policies*

**renderer:** networkd

**lom:**

**match:**

**driver:** ixgbe

*# you are responsible for setting tight enough match rules*

*# that only match one device if you use set-name*

**set-name:** lom1

**dhcp6:** true

**switchports:**

*# all cards on second PCI bus unconfigured by*

*# themselves, will be added to br0 below*

**match:**

**name:** enp2\*

**mtu:** 1280

**wifis:**

**all-wlans:**

*# useful on a system where you know there is*

*# only ever going to be one device*

**match:** **{}**

**access-points:**

"Joe's home"**:**

*# mode defaults to "infrastructure" (client)*

**password:** "s3kr1t"

*# this creates an AP on wlp1s0 using hostapd*

*# no match rules, thus the ID is the interface name*

**wlp1s0:**

**access-points:**

"guest"**:**

**mode:** ap

*# no WPA config implies default of open*

**bridges:**

*# the key name is the name for virtual (created) interfaces*

*# no match: and set-name: allowed*

**br0:**

*# IDs of the components; switchports expands into multiple interfaces*

**interfaces:** **[**wlp1s0**,** switchports**]**

**dhcp4:** true

How to enable DHCP on an interface

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

How to configure a static IP address on an interface

To set a static IP address, use the addresses keyword, which takes a list of (IPv4 or IPv6) addresses along with the subnet prefix length (e.g. /24).

**network:**

**version:** 2

**renderer:** networkd

**ethernets:**

**enp3s0:**

**addresses:**

**-** 10.10.10.2/24

How to configure DNS servers and search domains

The lists of search domains and DNS server IP addresses can be defined as below:

**network:**

**version:** 2

**renderer:** networkd

**ethernets:**

**enp3s0:**

**addresses:**

**-** 10.10.10.2/24

**nameservers:**

**search:**

**-** "mycompany.local"

**addresses:**

**-** 10.10.10.253

**-** 8.8.8.8

How to connect multiple interfaces with DHCP

DHCP can be used with multiple interfaces. The metrics for the routes acquired from DHCP can be changed with the use of DHCP overrides.

In this example, enp5s0 is preferred over enp6s0, as it has a lower route metric:

**network:**

**version:** 2

**ethernets:**

**enp5s0:**

**dhcp4:** yes

**dhcp4-overrides:**

**route-metric:** 100

**enp6s0:**

**dhcp4:** yes

**dhcp4-overrides:**

**route-metric:** 200

How to connect to an open wireless network

For open wireless networks, Netplan only requires that the access point is defined. In this example, opennetwork is the network SSID:

**network:**

**version:** 2

**wifis:**

**wl0:**

**access-points:**

**opennetwork:** **{}**

**dhcp4:** yes

How to configure your computer to connect to your home Wi-Fi network

If all you need is to connect to your local domestic Wi-Fi network, use the configuration below:

**network:**

**version:** 2

**renderer:** NetworkManager

**wifis:**

**wlp2s0b1:**

**dhcp4:** yes

**access-points:**

"network\_ssid\_name"**:**

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

How to connect to a WPA Personal wireless network without DHCP

For private wireless networks, the access point name and password must 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

How to connect to WPA Enterprise wireless networks with EAP+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

How to connect to WPA Enterprise wireless networks with EAP+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.readthedocs.io/en/stable/reference/) page.

How to use multiple addresses on a single interface

The addresses keyword can take a list of addresses to assign to an interface. You can also defined a label for each address:

**network:**

**version:** 2

**renderer:** networkd

**ethernets:**

**enp3s0:**

**addresses:**

**-** 10.100.1.37/24

**-** **10.100.1.38/24:**

**label:** "enp3s0:0"

**-** **10.100.1.39/24:**

**label:** "enp3s0:some-label"

How to use 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.

How to use NetworkManager as a renderer

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

**network:**

**version:** 2

**renderer:** NetworkManager

How to configure interface bonding

Bonding is configured by declaring a bond interface with a list of physical interfaces and a bonding mode:

**network:**

**version:** 2

**renderer:** networkd

**bonds:**

**bond0:**

**dhcp4:** yes

**interfaces:**

**-** enp3s0

**-** enp4s0

**parameters:**

**mode:** active-backup

**primary:** enp3s0

How to configure multiple bonds

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

How to configure network bridges

Use the following configuration to create a simple bridge consisting of a single device that uses DHCP:

**network:**

**version:** 2

**renderer:** networkd

**ethernets:**

**enp3s0:**

**dhcp4:** no

**bridges:**

**br0:**

**dhcp4:** yes

**interfaces:**

**-** enp3s0

How to create a bridge with a VLAN for libvirtd

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/libvirt/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>**

How to create VLANs

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 **]**

How to use 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 configuration would be very similar:

**network:**

**version:** 2

**renderer:** networkd

**ethernets:**

**ens3:**

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

**routes:**

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

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

**on-link:** true

How to configure source routing

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

How to configure 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:**

**addresses:** **[** "127.0.0.1/8"**,** "::1/128"**,** "7.7.7.7/32" **]**

How to integrate with Windows DHCP Server

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

**network:**

**version:** 2

**ethernets:**

**enp3s0:**

**dhcp4:** yes

**dhcp-identifier:** mac

How to connect to an IPv6 over IPv4 tunnel

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"

How to configure 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" **]**

How to connect two systems with a WireGuard VPN

Generate the private and public keys in the first peer. Run the following commands with administrator privileges:

*wg genkey > private.key*

*wg pubkey < private.key > public.key*

*cat private.key*

*UMjI9WbobURkCDh2RT8SRM5osFI7siiR/sPOuuTIDns=*

*cat public.key*

*EdNnZ1/2OJZ9HcScSVcwDVUsctCkKQ/xzjEyd3lZFFs=*

Do the same in the second peer:

*wg genkey > private.key*

*wg pubkey < private.key > public.key*

*cat private.key*

*UAmjvLDVuV384OWFJkmI4bG8AIAZAfV7LarshnV3+lc=*

*cat public.key*

*AIm+QeCoC23zInKASmhu6z/3iaT0R2IKraB7WwYB5ms=*

Use the following configuration in the first peer (replace the keys and IP addresses as needed):

**network:**

**tunnels:**

**wg0:**

**mode:** wireguard

**port:** 51820

**key:** UMjI9WbobURkCDh2RT8SRM5osFI7siiR/sPOuuTIDns=

**addresses:**

**-** 172.16.0.1/24

**peers:**

**-** **allowed-ips:** **[**172.16.0.0/24**]**

**endpoint:** 10.86.126.56:51820

**keys:**

**public:** AIm+QeCoC23zInKASmhu6z/3iaT0R2IKraB7WwYB5ms=

In the YAML file above, key is the first peer’s private key and public is the second peer’s public key. endpoint is the second peer IP address.

Use the following configuration in the second peer:

**network:**

**tunnels:**

**wg0:**

**mode:** wireguard

**port:** 51820

**key:** UAmjvLDVuV384OWFJkmI4bG8AIAZAfV7LarshnV3+lc=

**addresses:**

**-** 172.16.0.2/24

**peers:**

**-** **allowed-ips:** **[**172.16.0.0/24**]**

**endpoint:** 10.86.126.40:51820

**keys:**

**public:** EdNnZ1/2OJZ9HcScSVcwDVUsctCkKQ/xzjEyd3lZFFs=

In the YAML file above, key is the second peer’s private key and public is the first peer’s public key. endpoint is the first peer's IP address.

How to connect your home computer to a cloud instance with a WireGuard VPN

Follow the same steps from the previous how-to to generate the necessary keys.

The difference here is that your computer is likely behind one or more devices doing NAT so you probably don’t have a static public IP to use as endpoint in the remote system.

Use the following configuration in your computer:

**network:**

**tunnels:**

**wg0:**

**mode:** wireguard

**port:** 51821

**key:** UMjI9WbobURkCDh2RT8SRM5osFI7siiR/sPOuuTIDns=

**addresses:**

**-** 172.17.0.1/24

**peers:**

**-** **allowed-ips:** **[**172.17.0.0/24**]**

**endpoint:** 54.234.x.y:51821

**keys:**

**public:** AIm+QeCoC23zInKASmhu6z/3iaT0R2IKraB7WwYB5ms=

Again, key is your private key and public is the remote system’s public key. The endpoint is the public IP address of your instance.

In the remote instance you just need to omit the endpoint.

**network:**

**tunnels:**

**wg0:**

**mode:** wireguard

**port:** 51821

**key:** UAmjvLDVuV384OWFJkmI4bG8AIAZAfV7LarshnV3+lc=

**addresses:**

**-** 172.17.0.2/24

**peers:**

**-** **allowed-ips:** **[**172.17.0.0/24**]**

**keys:**

**public:** EdNnZ1/2OJZ9HcScSVcwDVUsctCkKQ/xzjEyd3lZFFs=

Don’t forget to allow the UDP port 51821 in your instance’s security group.

After applying your configuration you should be able to reach your remote instance through the IP address 172.17.0.2.