NIC Configuration

IP command and Netplan

Background Information

Things you need to know

Networking Refresher

- OSI Layered Architecture
- For list of supported protocols in your Linux OS see:
  - /etc/protocols
- IETF is the sole publisher of protocols since 1986
  - TCP/IP was started by US Department of Defense in the 1960's
  - TCP/IP is an open standard (anyone can use it)
  - There were other "commercial" network communications that did not last
    - Appletalk, NCP, IPX, SPX, NetBEUI, UUCP

#	Layer Name	PDU	Description	Example of Protocols
7	Application	Data	Network Process to application	DNS, FTP, HTTP, SMTP, Telnet, DHCP
6	Presentation	Data	Data representation and encryption	GIF, JPEG, SSL, MIME
5	Session	Data	Interhost communication	NetBIOS, Sockets, Named Pipes, RPC
4	Transport	Segments	End-to-end connections and reliability	TCP, UDP
3	Network	Packets	Path determination	IPv4, IPv6, IPSec, BGP, ICMP, IGMP
2	Datalink	Frames	Physical addressing	Ethernet, MAC, WIFI, LLC
1	Physical	Bits	Medium, signal, and binary transmission	Cat5e, Fiber, Wireless

L3: IP/sub, Gateway, DNS

L2: Link Speed, UP/Down, SSID, Encryption(wifi)

Networking on Server vs. Client

- Servers may have many interfaces depending on their role
- Generally we want the networks to be static and stay the same after each boot-up
- May have to deal with many protocols and complex configs (vlans, bgp, static routes, vpn, Ipsec, etc.)

- Clients (Laptop or Desktop) have one or two interfaces (ex. LAN and WLAN)
- The clients usually have dynamic configurations and may/should not retain that config after each boot-up
  - Have to dynamically or automatically adjust (DHCP, WIFI, etc)

Network Renderer in Linux

- NIC device driver in the Linux Kernel provides communication to the network hardware via a device file
  - For example your browser reads and writes to a /dev/eth0 file when communicating to internet
- Network Renderers are daemons that are used to manage network configurations
  - Linux systemd (systemd-networkd)
  - Network Manager daemon
- You can configure or use scripts to work directly with the network renderer on the system
- Usually we use a front-end utility to do this

Systemd-networkd

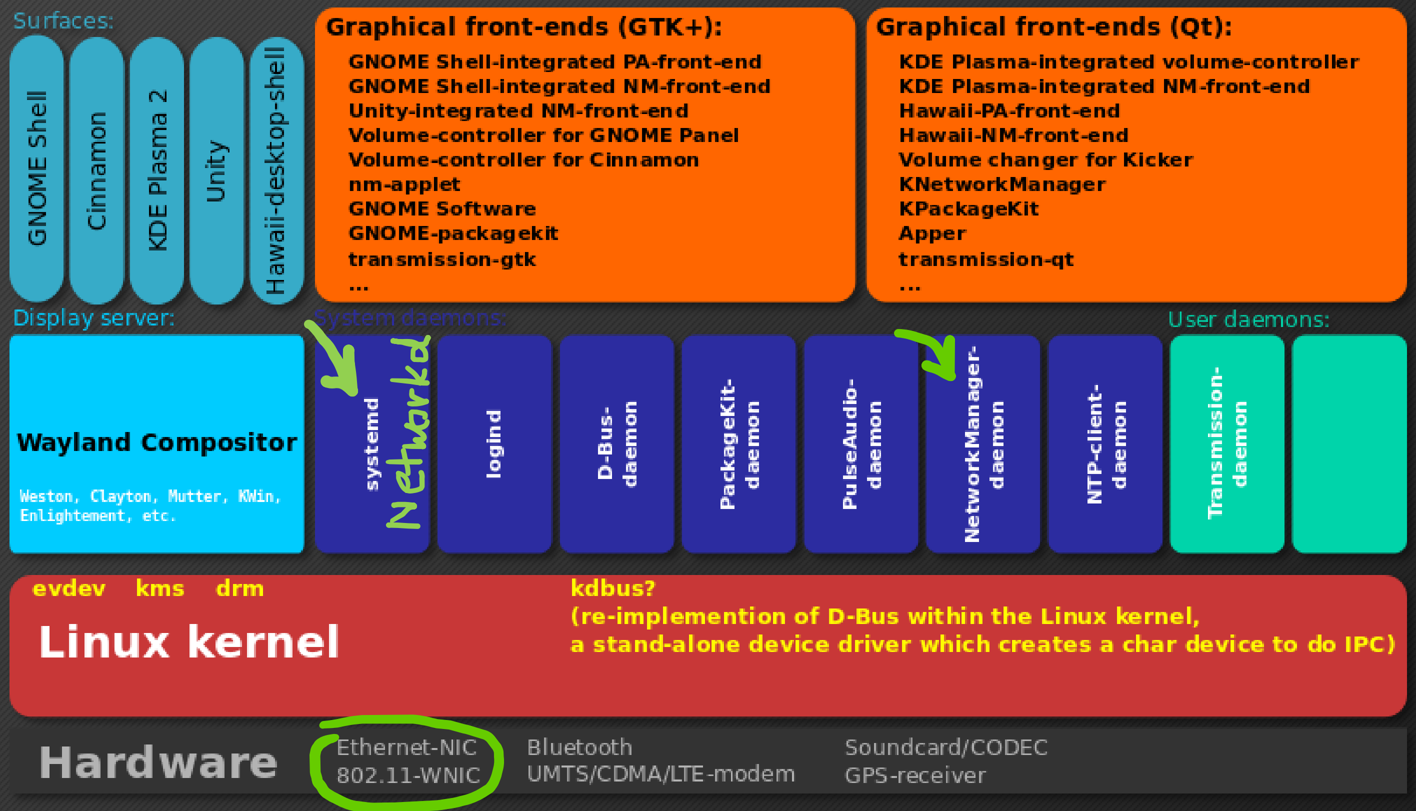
- Networkd is a network renderer in Linux that comes bundled with systemd
- It uses ini-style files to initiate network settings
- Directly reads configurations files on Startup
- Configuration files may be (depending on the distro) located at:
  - /etc/network/ (check out these folders in your VM!)
  - /etc/systemd/network
  - /usr/lib/systemd/network
  - /usr/local/lib/systemd/network

```
Sample Config File content:
  [Match]
  Name=ens33

  [Network]
  Address=192.168.1.20/24
  Gateway=192.168.1.1
  DNS=192.168.1.1
```

Network Manager

- Developed by Red Hat in 2004 to deal with “modern” networking needs.
- Default config file:
  - /etc/NetworkManager/NetworkManager.conf
    - (is this present in your Ubuntu Server?)
- Many desktop distributions include:
  - Network Manager Daemon
  - GUI front end



Surfaces:

GNOME Shell

Cinnamon

KDE Plasma 2

Unity

Hawaii-desktop-shell

Graphical front-ends (GTK+):

GNOME Shell-integrated PA-front-end
GNOME Shell-integrated NM-front-end
Unity-integrated NM-front-end
Volume-controller for GNOME Panel
Volume-controller for Cinnamon
nm-applet
GNOME Software
GNOME-packagekit
transmission-gtk
...

Graphical front-ends (Qt):

KDE Plasma-integrated volume-controller
KDE Plasma-integrated NM-front-end
Hawaii-PA-front-end
Hawaii-NM-front-end
Volume changer for Kicker
KNetworkManager
KPackageKit
Apper
transmission-qt
...

Display server:

Wayland Compositor

Weston, Clayton, Mutter, KWin, Enlightement, etc.

System daemons:

systemd

Networkd

logind

D-Bus-daemon

PackageKit-daemon

PulseAudio-daemon

NetworkManager-daemon

NTP-client-daemon

User daemons:

Transmission-daemon

evdev kms drm

Linux kernel

kdbus?
(re-implemention of D-Bus within the Linux kernel,
a stand-alone device driver which creates a char device to do IPC)

Hardware

Ethernet-NIC
802.11-WNIC

Bluetooth
UMTS/CDMA/LTE-modem

Soundcard/CODEC
GPS-receiver

Distros for every need!

- Want a Router? NAS, or Firewall Appliance? IoT? IoS? There is a distro for that!
  - Every distro may come with different way of doing networking depending on the goals and the target market
- Distros have different software options and combinations to choose from for networking needs (both for the back-end and the front-end)
- Read the documentation for your distro to find out how the networking configuration is accomplished in your specific distro
  - For Ubuntu Server: https://ubuntu.com/server/docs/network-configuration

Network Config in GUI

- Different distros use different GUI environments that come with different network configuration tools
  - Suse has YaST, Others (including Ubuntu) may use GNOME, KDE, or other GUI envinments
  - GUI tools are ideal for configuring Wifi on a laptop on the go
- All these different GUI tools may/can use different network renderers
- If used incorrectly, settings may conflict with, or override the configuration files and commands that were entered by you!
- Read the Distros documentation!
  - For Ubuntu Desktop see here: https://help.ubuntu.com/stable/ubuntu-help/net.html.en

Network Configuration in CLI

- Instead of working with Network Renderer directly we may use various utilities to achieve the configurations we need
  - Ifconfig
  - ip
  - netplan
  - ethtool
  - etc…
- Different distros may come with different tools, but for the most part these utilities have been around for a long time and commands are standard
- Ubuntu 20.04 and many distros have deprecated ifconfig and have switched to more modern, more powerful ip and netplan
  - You can also choose to use networkd or Network Manager in Ubuntu

Configuring NIC in CLI: IP Command

WARNING: Changes may or may not persist!

Meet the IP Command

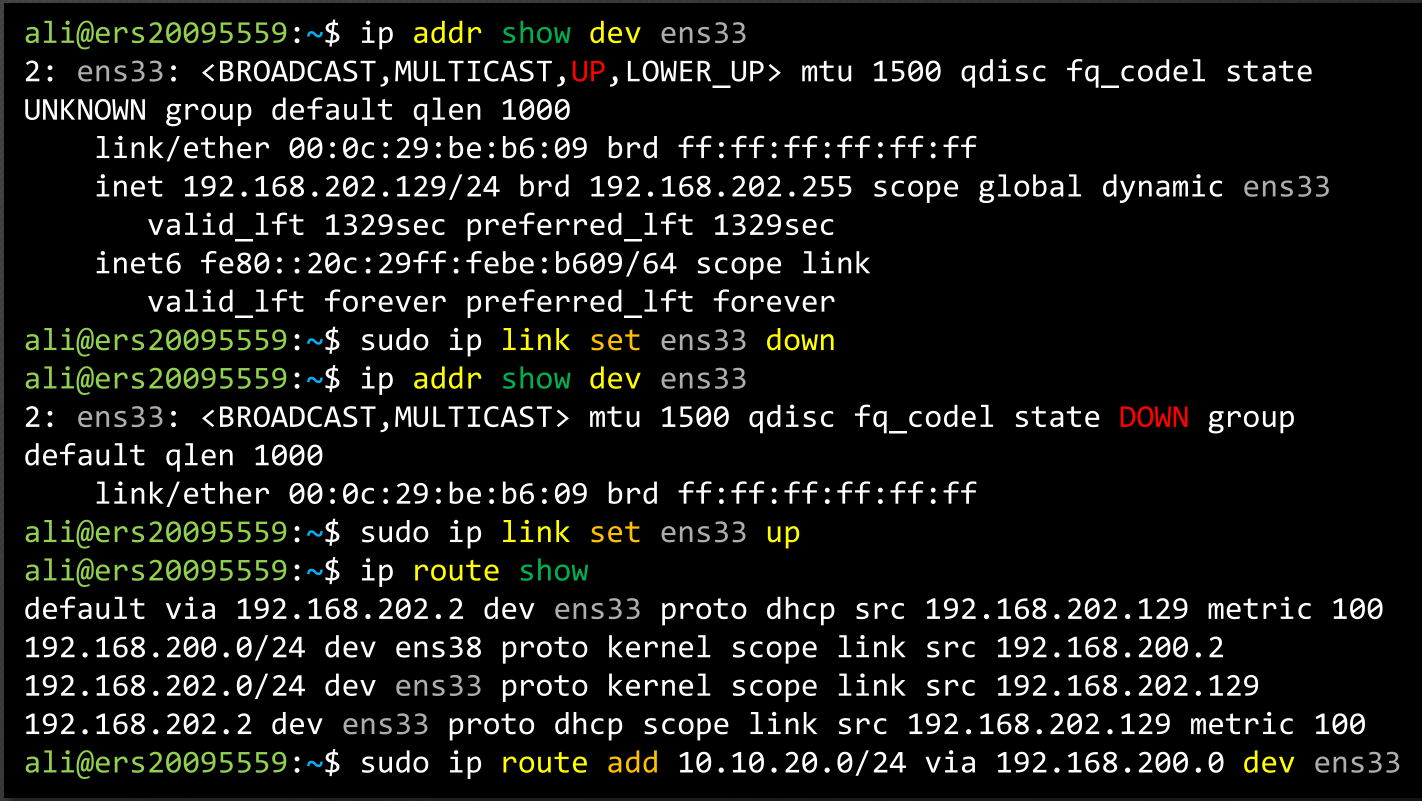
- “ifconfig” is deprecated (you can still install ifconfig manually Ubuntu)
- IP command is designed to be one command to rule them all!
- Functionally organized on Layer 2 and 3 of the network stack
- Capable of almost all networking related tasks:
  - Displaying or Modifying Interface properties
  - Adding, Removing ARP Cache entries along creating new Static ARP entry for a host.
  - Displaying MAC addresses associated with all the interfaces.
  - Displaying and modifying kernel routing tables (route).

IP Command in Ubuntu Server

- Read the Server Guide under IP Addressing section:
  - https://ubuntu.com/server/docs/network-configuration
- Configuration changes using IP commands take effect immediately
- The configurations are NOT persistent in Ubuntu server (They are lost after a reboot)
- For persistent settings use Netplan instead!

IP Command Structure

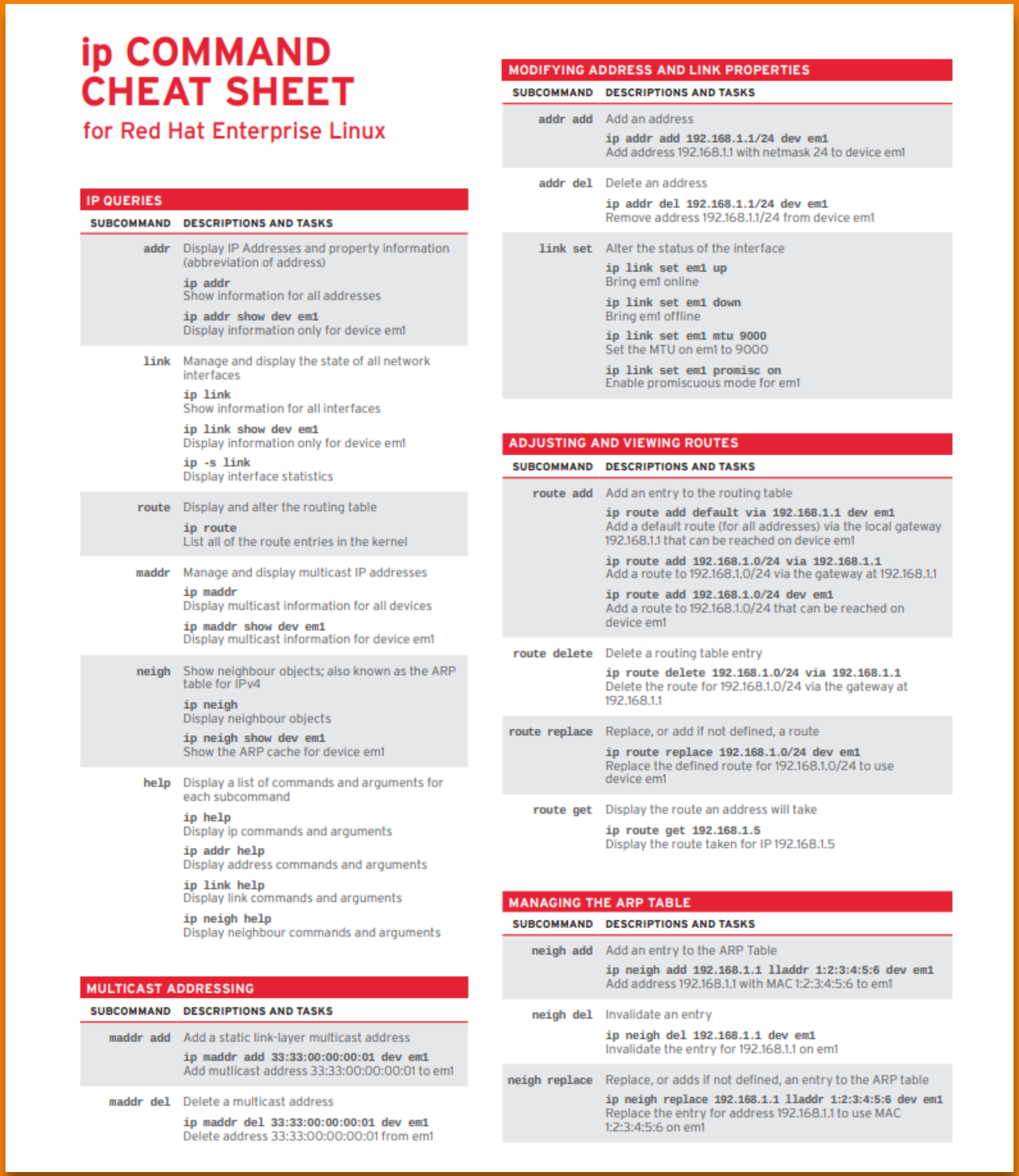
- Call the IP command (i.e. `ip` or `sudo ip`)
- Then use sub-commands to target protocol/layer:
  - `address` (Layer 3)
  - `link` (layer 2, physical interface itself)
  - `maddr` (layer 3, Multi-cast address)
  - `neigh` (layer 2, ARP tables)
  - `route` (layer 3)
- Then use show or modifying sub-commands:
  - `show`
  - `add`, `del`, `set`
- Target specific device or interface
  - `dev`



```
ali@ers20095559:~$ ip addr show dev ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UNKNOWN group default qlen 1000
    link/ether 00:0c:29:be:b6:09 brd ff:ff:ff:ff:ff:ff
    inet 192.168.202.129/24 brd 192.168.202.255 scope global dynamic ens33
       valid_lft 1329sec preferred_lft 1329sec
    inet6 fe80::20c:29ff:febe:b609/64 scope link
       valid_lft forever preferred_lft forever
ali@ers20095559:~$ sudo ip link set ens33 down
ali@ers20095559:~$ ip addr show dev ens33
2: ens33: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel state DOWN group
default qlen 1000
    link/ether 00:0c:29:be:b6:09 brd ff:ff:ff:ff:ff:ff
ali@ers20095559:~$ sudo ip link set ens33 up
ali@ers20095559:~$ ip route show
default via 192.168.202.2 dev ens33 proto dhcp src 192.168.202.129 metric 100
192.168.200.0/24 dev ens38 proto kernel scope link src 192.168.200.2
192.168.202.0/24 dev ens33 proto kernel scope link src 192.168.202.129
192.168.202.2 dev ens33 proto dhcp scope link src 192.168.202.129 metric 100
ali@ers20095559:~$ sudo ip route add 10.10.20.0/24 via 192.168.200.0 dev ens33
```

IP Command Cheat Sheet

- Here is a nice IP Command Cheat Sheet from RedHat: https://access.redhat.com/sites/default/files/attachments/rh_ip_command_cheatsheet_1214_jcs_print.pdf



ip COMMAND CHEAT SHEET

for Red Hat Enterprise Linux

IP QUERIES

SUBCOMMAND	DESCRIPTIONS AND TASKS

`addr`	Display IP Addresses and property information (abbreviation of address)
	`ip addr`
	Show information for all addresses
	`ip addr show dev em1`
	Display information only for device em1

`link`	Manage and display the state of all network interfaces
	`ip link`
	Show information for all interfaces
	`ip link show dev em1`
	Display information only for device em1
	`ip -s link`
	Display interface statistics

`route`	Display and alter the routing table `ip route` List all of the route entries in the kernel

`maddr`	Manage and display multicast IP addresses
	`ip maddr`
	Display multicast information for all devices
	`ip maddr show dev em1`
	Display multicast information for device em1

`neigh`	Show neighbour objects; also known as the ARP table for IPv4
	`ip neigh`
	Display neighbour objects
	`ip neigh show dev em1`
	Show the ARP cache for device em1

`help`	Display a list of commands and arguments for each subcommand
	`ip help`
	Display ip commands and arguments
	`ip addr help`
	Display address commands and arguments
	`ip link help`
	Display link commands and arguments
	`ip neigh help`
	Display neighbour commands and arguments

MULTICAST ADDRESSING

SUBCOMMAND	DESCRIPTIONS AND TASKS
`maddr add`	Add a static link-layer multicast address `ip maddr add 33:33:00:00:00:01 dev em1` Add mutlicast address 33:33:00:00:00:01 to em1
`maddr del`	Delete a multicast address `ip maddr del 33:33:00:00:00:01 dev em1` Delete address 33:33:00:00:00:01 from em1

MODIFYING ADDRESS AND LINK PROPERTIES

SUBCOMMAND	DESCRIPTIONS AND TASKS
`addr add`	Add an address `ip addr add 192.168.1.1/24 dev em1` Add address 192.168.1.1 with netmask 24 to device em1
`addr del`	Delete an address `ip addr del 192.168.1.1/24 dev em1` Remove address 192.168.1.1/24 from device em1

`link set`	Alter the status of the interface
	`ip link set em1 up`
	Bring em1 online
	`ip link set em1 down`
	Bring em1 offline
	`ip link set em1 mtu 9000`
	Set the MTU on em1 to 9000
	`ip link set em1 promisc on`
	Enable promiscuous mode for em1

ADJUSTING AND VIEWING ROUTES

SUBCOMMAND	DESCRIPTIONS AND TASKS

`route add`	Add an entry to the routing table
	`ip route add default via 192.168.1.1 dev em1`
	Add a default route (for all addresses) via the local gateway 192.168.1.1 that can be reached on device em1
	`ip route add 192.168.1.0/24 via 192.168.1.1`
	Add a route to 192.168.1.0/24 via the gateway at 192.168.1.1
	`ip route add 192.168.1.0/24 dev em1`
	Add a route to 192.168.1.0/24 that can be reached on device em1

`route delete`	Delete a routing table entry `ip route delete 192.168.1.0/24 via 192.168.1.1` Delete the route for 192.168.1.0/24 via the gateway at 192.168.1.1
`route replace`	Replace, or add if not defined, a route `ip route replace 192.168.1.0/24 dev em1` Replace the defined route for 192.168.1.0/24 to use device em1
`route get`	Display the route an address will take `ip route get 192.168.1.5` Display the route taken for IP 192.168.1.5

MANAGING THE ARP TABLE

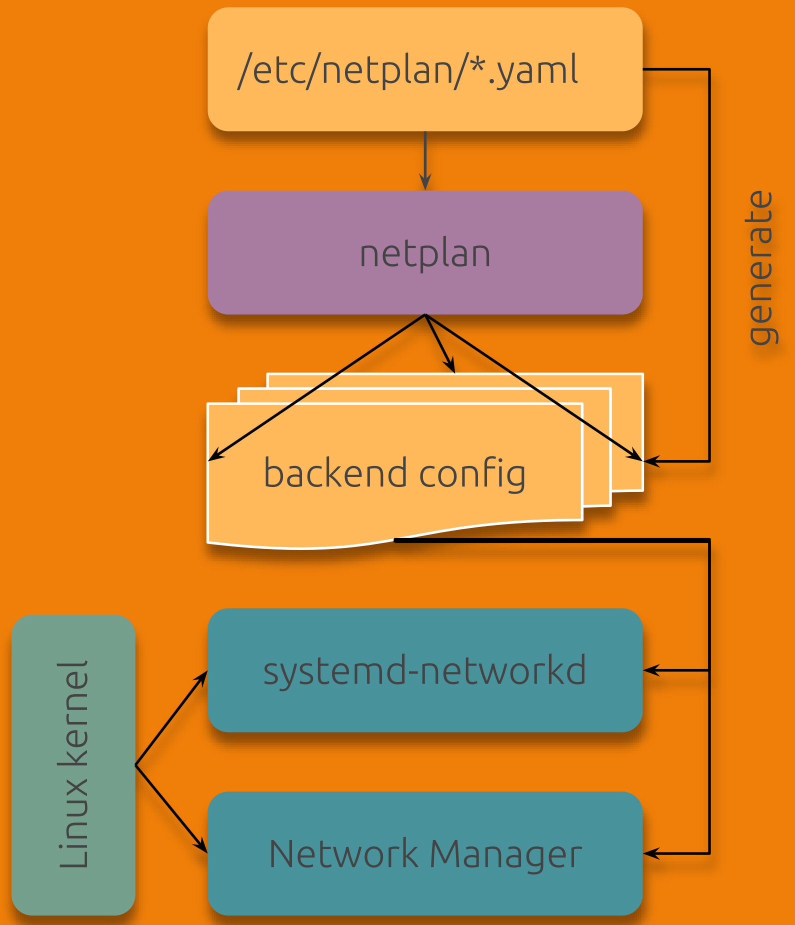
SUBCOMMAND	DESCRIPTIONS AND TASKS
`neigh add`	Add an entry to the ARP Table `ip neigh add 192.168.1.1 lladdr 1:2:3:4:5:6 dev em1` Add address 192.168.1.1 with MAC 1:2:3:4:5:6 to em1
`neigh del`	Invalidate an entry `ip neigh del 192.168.1.1 dev em1` Invalidate the entry for 192.168.1.1 on em1
`neigh replace`	Replace, or adds if not defined, an entry to the ARP table `ip neigh replace 192.168.1.1 lladdr 1:2:3:4:5:6 dev em1` Replace the entry for address 192.168.1.1 to use MAC 1:2:3:4:5:6 on em1

Persistent configuration: Netplan

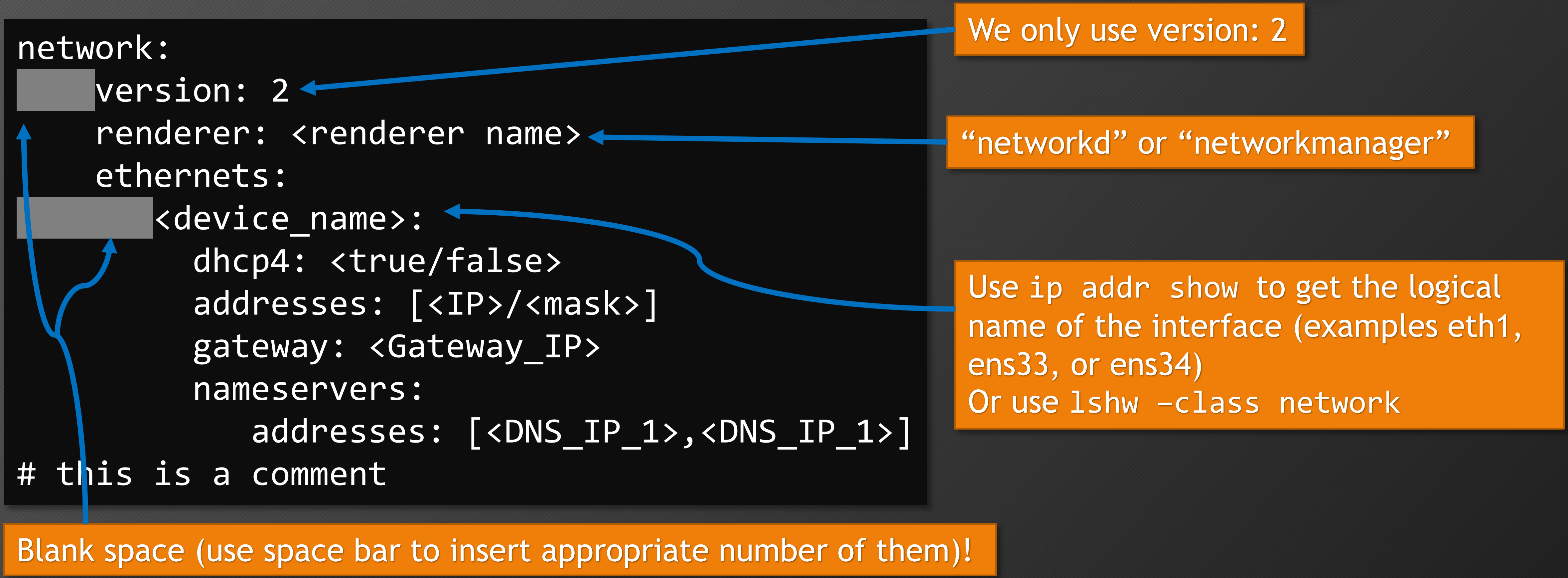
Debug that YAML!

Introducing netplan

- Visit this site: https://netplan.io/
- Uses YAML format for configuration files
- Reads configurations from files located at /etc/netplan/*.yaml
- Applies the setting from the YAML files one after another (numerically in order of file names) to the network renderer in use by the system during boot
  - Currently Netplan supports Network Manager and system-networkd



Netplan YAML files



```
network:
    version: 2
    renderer: <renderer name>
    ethernets:
       <device_name>:
         dhcp4: <true/false>
         addresses: [<IP>/<mask>]
         gateway: <Gateway_IP>
         nameservers:
            addresses: [<DNS_IP_1>,<DNS_IP_1>]
# this is a comment
```

We only use version: 2

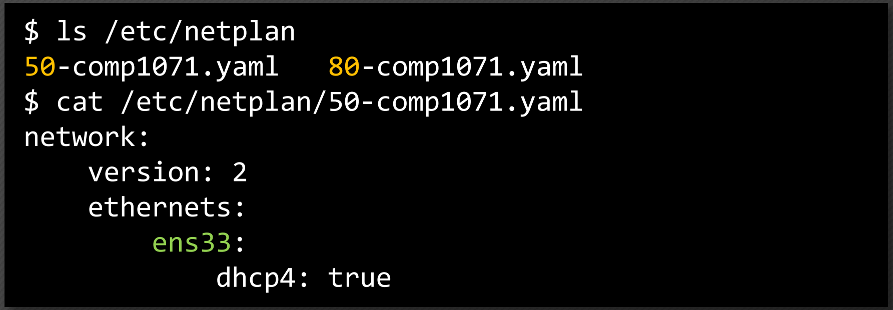
“networkd” or “networkmanager”

Use `ip addr show` to get the logical name of the interface (examples eth1, ens33, or ens34)
Or use `lshw -class network`

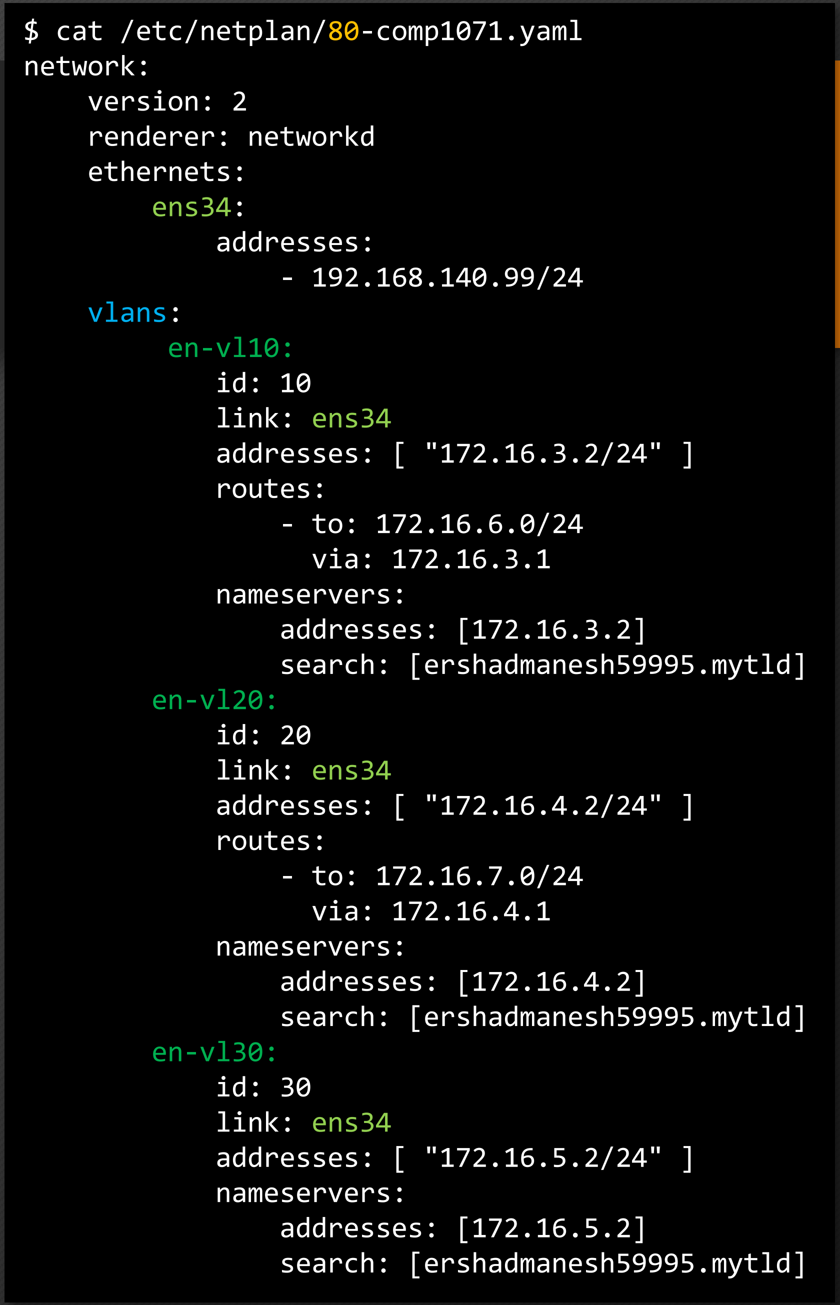
Blank space (use space bar to insert appropriate number of them)!

Example YAML file

- This is an examples of netplan .yaml files
- We have two NICs on the server. Since our configuration is more complex we decided to break the configuration into two separate files (one for each NIC)
  - 50-comp1071.yaml
  - 80-comp1071.yaml
- Netplan reads through each .yaml file numerically and applies the settings
  - If an interface is repeated, then configuration in the last file will override the previous configuration



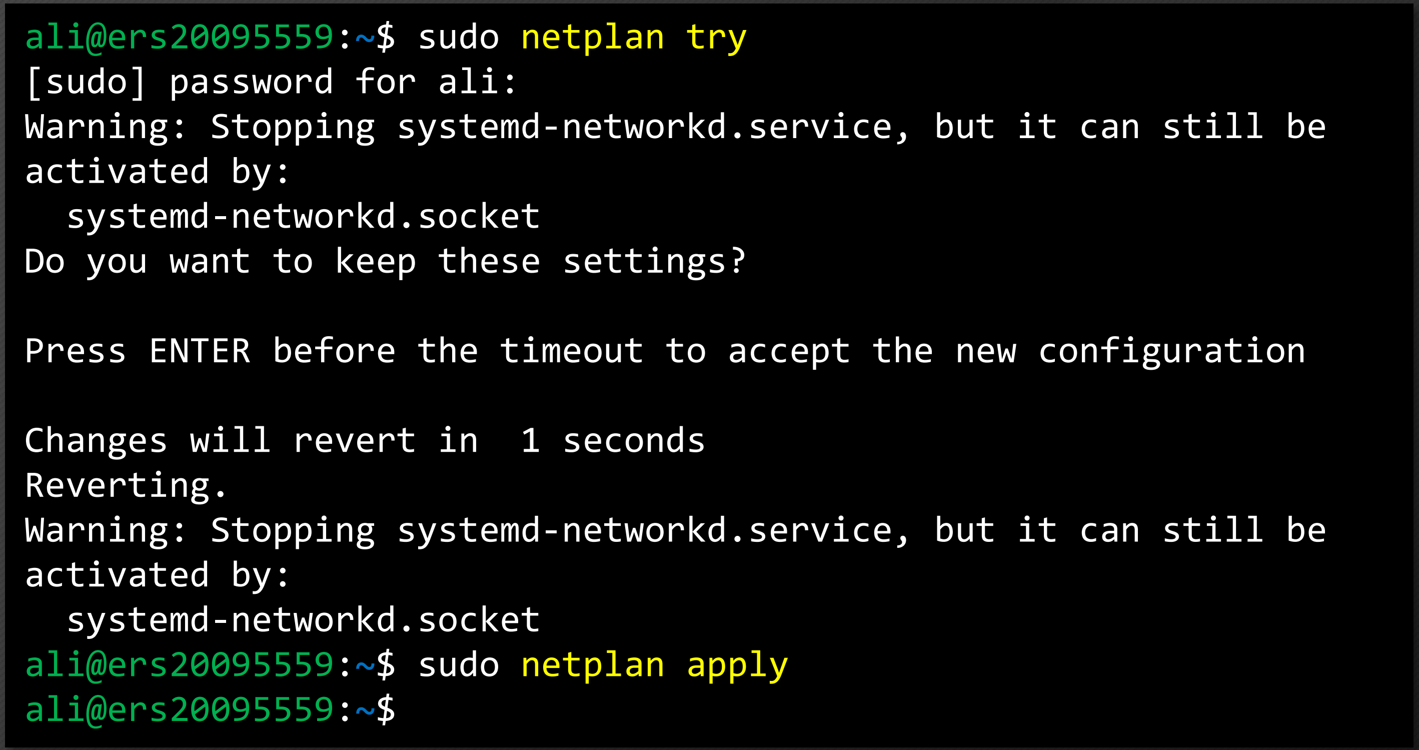
```
$ ls /etc/netplan
50-comp1071.yaml   80-comp1071.yaml
$ cat /etc/netplan/50-comp1071.yaml
network:
    version: 2
    ethernets:
        ens33:
            dhcp4: true
```



```
$ cat /etc/netplan/80-comp1071.yaml
network:
    version: 2
    renderer: networkd
    ethernets:
        ens34:
            addresses:
                - 192.168.140.99/24
    vlans:
         en-vl10:
            id: 10
            link: ens34
            addresses: [ "172.16.3.2/24" ]
            routes:
                - to: 172.16.6.0/24
                  via: 172.16.3.1
            nameservers:
                addresses: [172.16.3.2]
                search: [ershadmanesh59995.mytld]
        en-vl20:
            id: 20
            link: ens34
            addresses: [ "172.16.4.2/24" ]
            routes:
                - to: 172.16.7.0/24
                  via: 172.16.4.1
            nameservers:
                addresses: [172.16.4.2]
                search: [ershadmanesh59995.mytld]
        en-vl30:
            id: 30
            link: ens34
            addresses: [ "172.16.5.2/24" ]
            nameservers:
                addresses: [172.16.5.2]
                search: [ershadmanesh59995.mytld]
```

Try and apply your settings

- `# netplan try`
  - Checks configuration file for errors
  - If no errors found then applies changes for 120 sec
  - If you do not confirm the settings within 120 sec the changes will revert back
    - This is good in case you are working on a server remotely and sever your network connection to server by accident!
- `# netplan apply`
  - makes changes permanent



```
ali@ers20095559:~$ sudo netplan try
[sudo] password for ali:
Warning: Stopping systemd-networkd.service, but it can still be
activated by:
  systemd-networkd.socket
Do you want to keep these settings?

Press ENTER before the timeout to accept the new configuration

Changes will revert in  1 seconds
Reverting.
Warning: Stopping systemd-networkd.service, but it can still be
activated by:
  systemd-networkd.socket
ali@ers20095559:~$ sudo netplan apply
ali@ers20095559:~$
```