Basics of Linux Net

Module Linux Networking

Serhii Zakharchenko

Module overview

Lections topics

Lection 1

- VirtualBox Networking
- Linux interfaces configuration
- DHCP server install and config

Lection 2

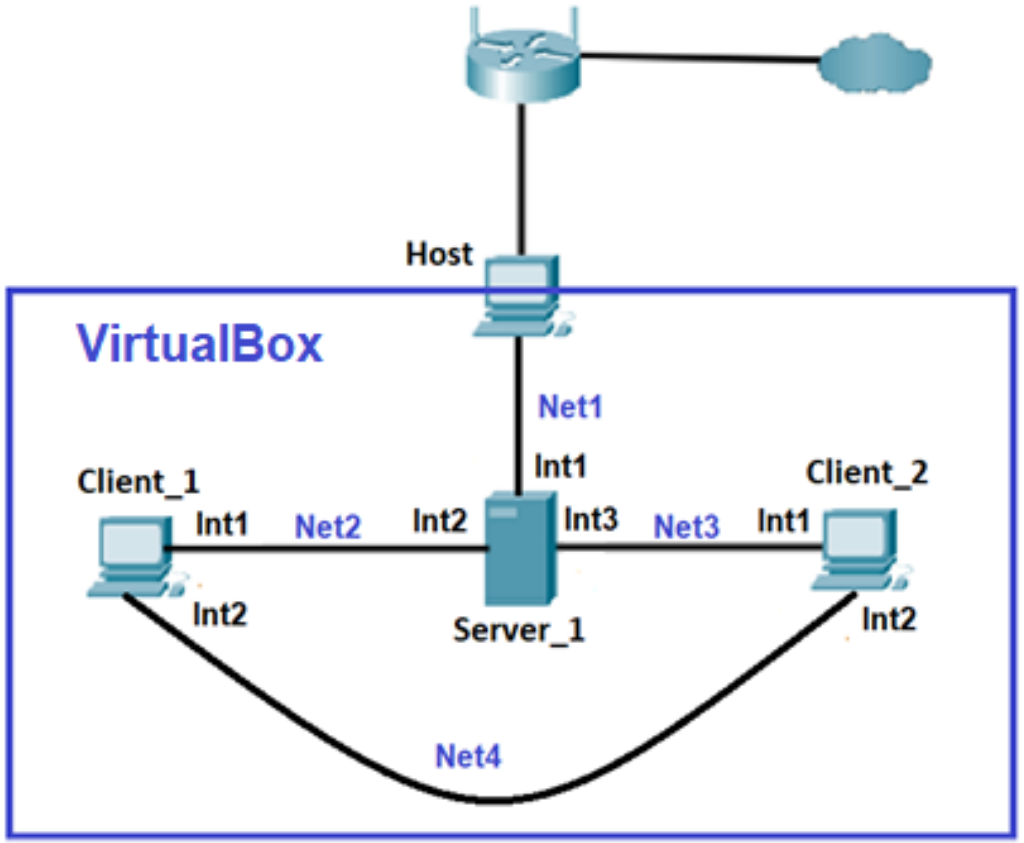
- IP routing configuration
- General troubleshooting procedures
- Linux monitoring and troubleshooting tools

Lection 3

- Remote access configuration
- Firewall technology overview
- Linux Firewall configuration

Practical task

- Server static IP-address configuration for Int1 and Int2
- Server DHCP client configuration for Int3
- DHCP Server config
- DHCP client config for Client_1 and Client_2
- SSH config
- Static routing config
- Traffic filtering config
- NAT config



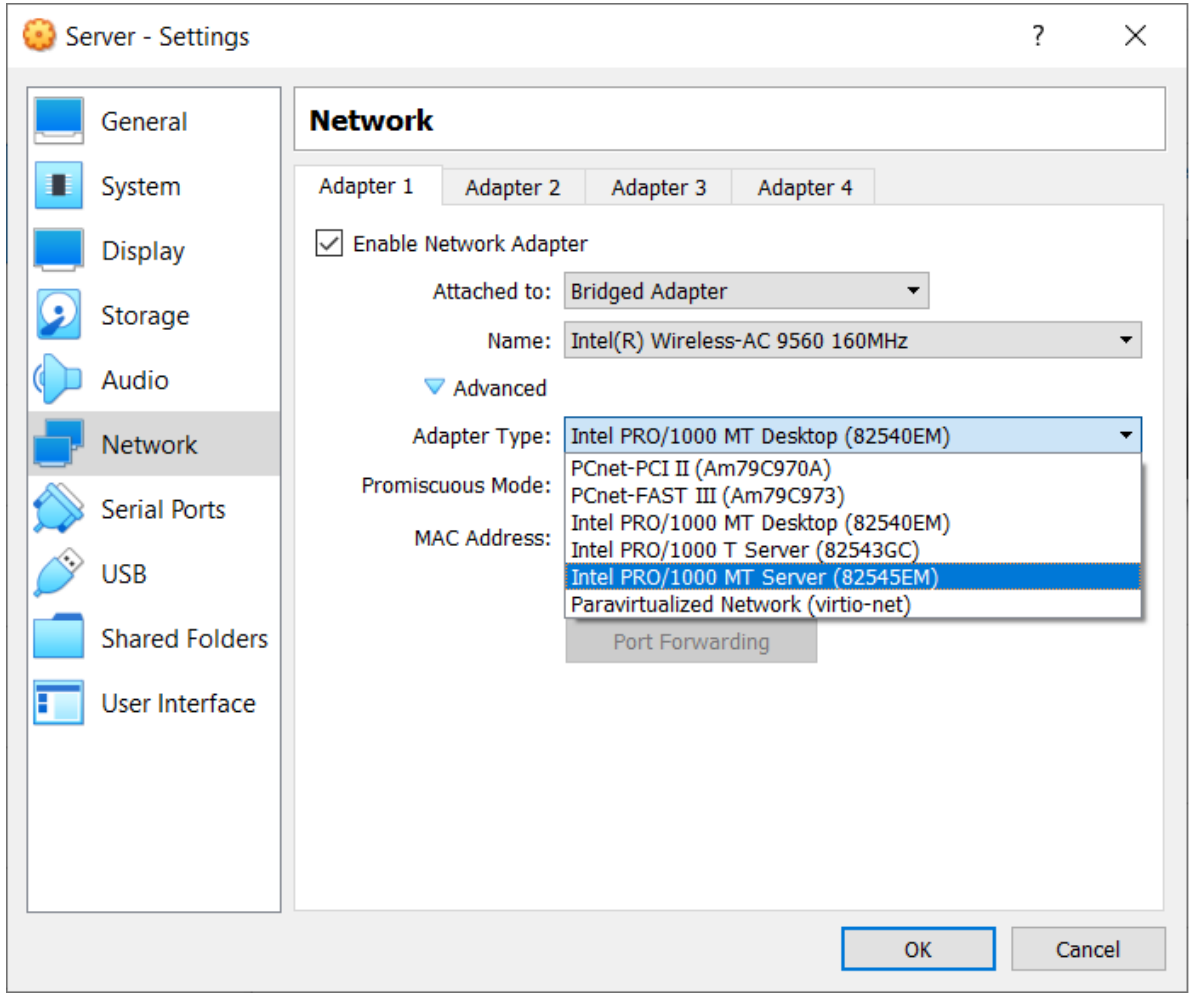
Agenda

- VirtualBox Networking
- Linux interfaces configuration
- Linux IP address config
- DNS client configuration
- DHCP server install and config
- Q&A

VirtualBox Networking

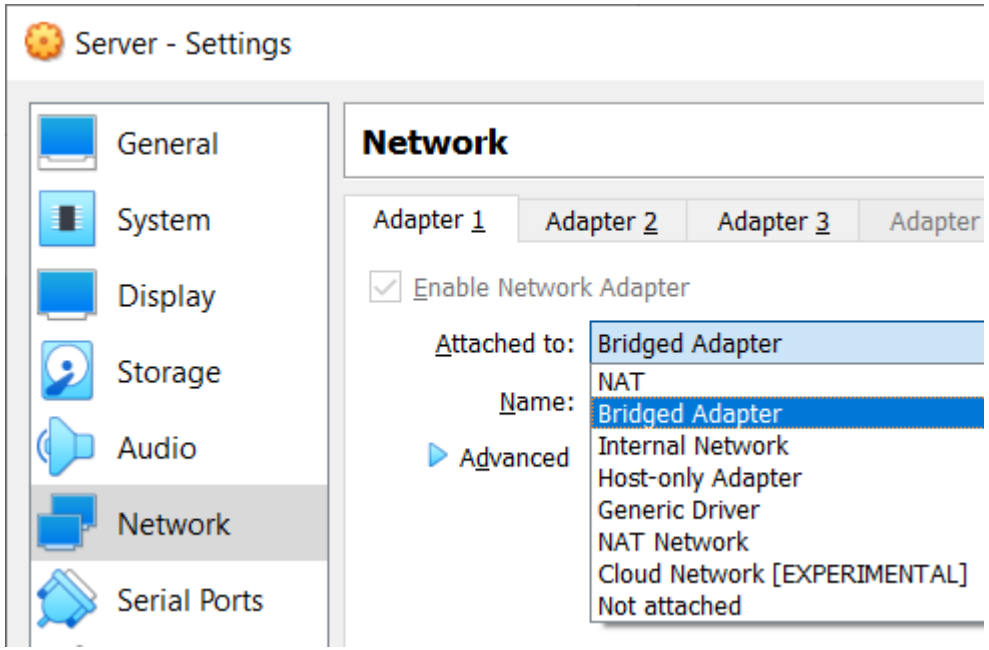
Virtual Networking

- Oracle VM VirtualBox provides up to **eight** virtual Network adapters for each virtual machine. For each such card, you can individually select the following:
  - The hardware that will be virtualized.
  - The virtualization mode that the virtual card operates in, with respect to your physical networking hardware on the host.
- **Four** of the network cards can be configured in the Network section of the Settings dialog in the graphical user interface of Oracle VM VirtualBox.
- You can configure all eight network cards on the command line using *VBoxManage modifyvm*.



Introduction to Networking Modes

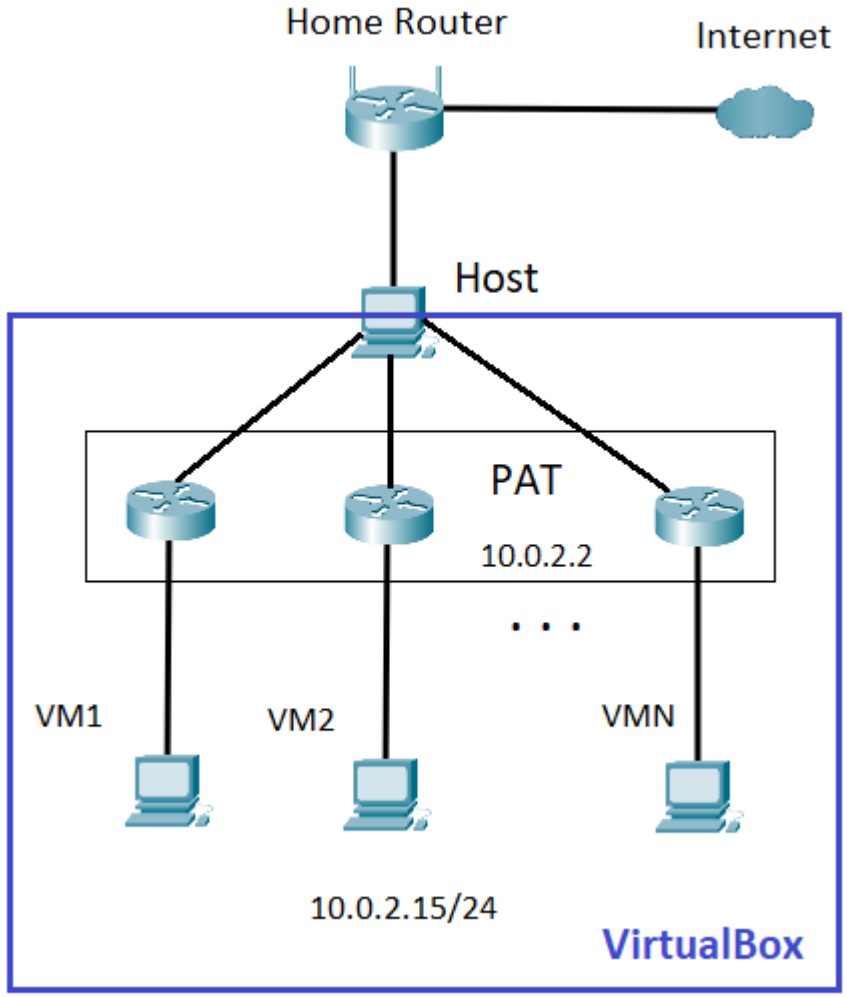
- Network Address Translation (NAT)
- Bridged networking
- Internal networking
- Host-only networking
- Generic networking
- NAT Network
- Not attached



Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	–	–
Internal	–	–	+	–	–
Bridged	+	+	+	+	+
NAT	+	Port forward	–	+	Port forward
NATservice	+	Port forward	+	+	Port forward

Network Address Translation (NAT)

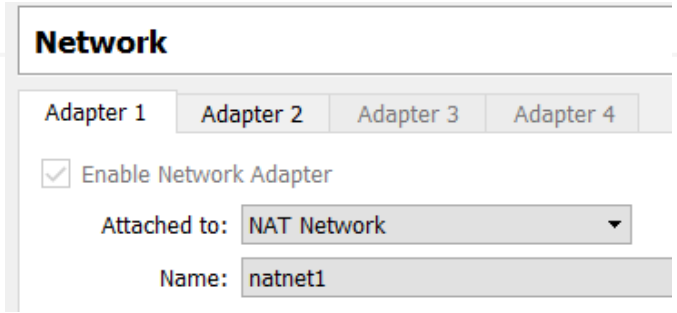
- Network Address Translation (NAT) is the simplest way of accessing an external network from a virtual machine. Usually, it **does not require any configuration** on the host network and guest system. It is the default networking mode in Oracle VM VirtualBox.
- A virtual machine with NAT enabled acts much like a real computer that connects to the Internet through a router. **The router, in this case, is the Oracle VM VirtualBox networking engine**, which maps traffic from and to the virtual machine transparently.
- In Oracle VM VirtualBox this **router is placed between each virtual machine and the host**. This separation maximizes security since by default virtual machines cannot talk to each other.

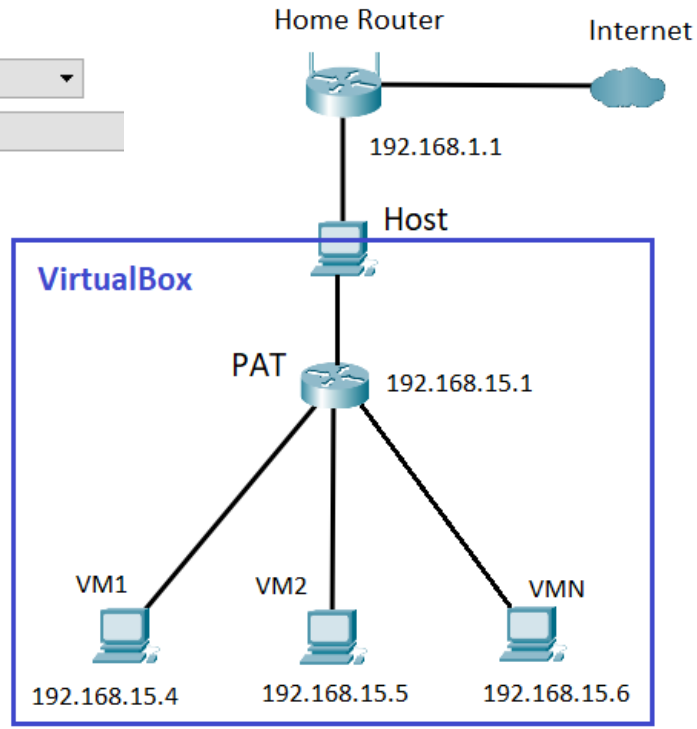


Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	–	–
Internal	–	–	+	–	–
Bridged	+	+	+	+	+
NAT	+	Port forward	–	+	Port forward
NATservice	+	Port forward	+	+	Port forward

NAT Network

- The Network Address Translation (NAT) service works in a similar way to a **home router**, grouping the systems using it into a network and preventing systems outside of this network from directly accessing systems inside it, but letting systems inside **communicate with each other** and with systems outside.
- A NAT service is attached to an internal network. Virtual machines which are to make use of it should be attached to that internal network.
- The name of internal network is chosen when the NAT service is created, and the internal network will be created if it does not already exist.
- The following is an example command to create a NAT network: ***VBoxManage natnetwork add --netname natnet1 --network "192.168.15.0/24" --enable***

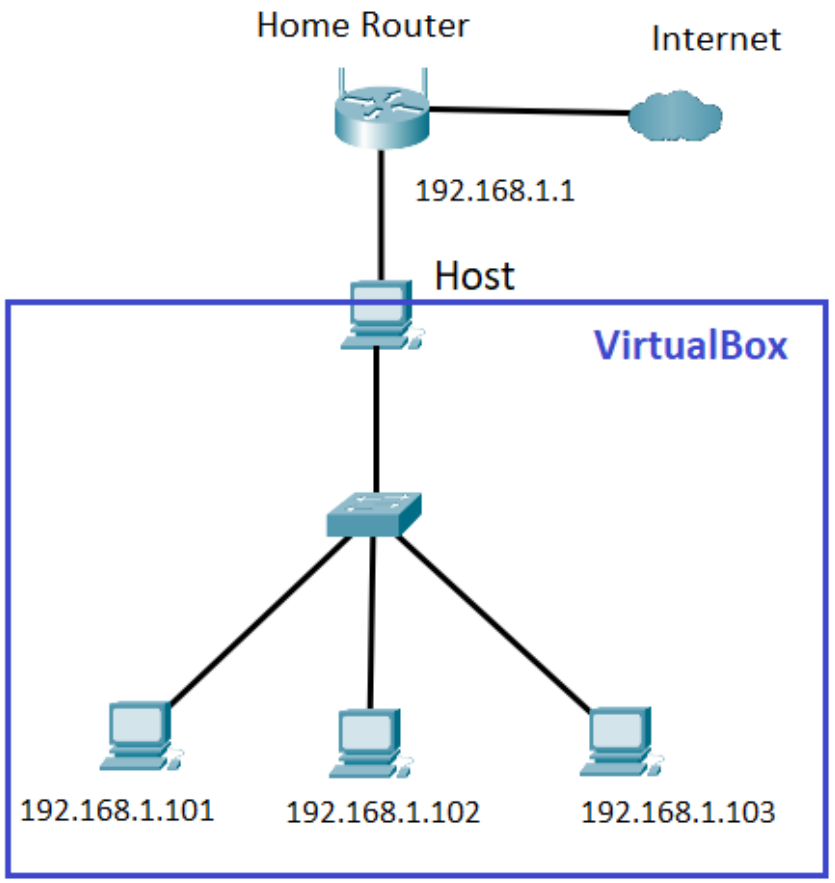




Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	–	–
Internal	–	–	+	–	–
Bridged	+	+	+	+	+
NAT	+	Port forward	–	+	Port forward
NATservice	+	Port forward	+	+	Port forward

Bridged networking

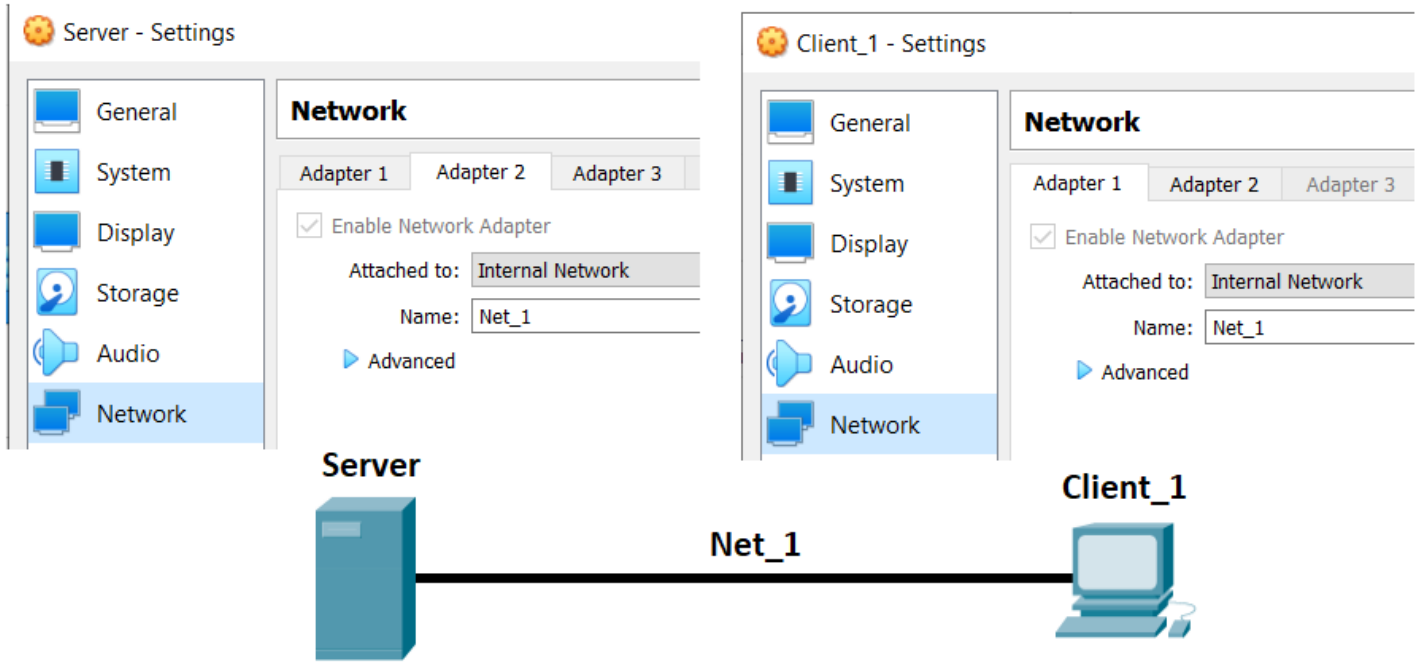
- With bridged networking, Oracle VM VirtualBox uses a device driver on your host system that filters data from your physical network adapter.
- This driver is therefore called a **net filter driver.** This enables Oracle VM VirtualBox to intercept data from the physical network and inject data into it, effectively creating a new network interface in software.
- When a guest is using such a new software interface, it looks to the host system as though the guest were **physically connected to the interface using a network cable**.
- The host can send data to the guest through that interface and receive data from it.



Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	–	–
Internal	–	–	+	–	–
Bridged	+	+	+	+	+
NAT	+	Port forward	–	+	Port forward
NATservice	+	Port forward	+	+	Port forward

Internal networking

- Internal Networking is similar to bridged networking in that the VM can directly communicate with the outside world. However, the **outside world is limited to other VMs** on the same host which connect to the same internal network.
- Every internal network is identified simply **by its name**.
- Once there is more than one active virtual network card with the same internal network ID, the Oracle VM VirtualBox support driver will automatically wire the cards and act as a **network switch**.
- The Oracle VM VirtualBox support driver implements a complete Ethernet switch and supports both broadcast/multicast frames and promiscuous mode.



Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	–	–
Internal	–	–	+	–	–
Bridged	+	+	+	+	+
NAT	+	Port forward	–	+	Port forward
NATservice	+	Port forward	+	+	Port forward

Linux interfaces configuration

Identify Ethernet Interfaces

To quickly identify all available Ethernet interfaces, you can use the commands:

$ ls /sys/class/net

```
sergey@Server1:~$ ls /sys/class/net
enp0s3  enp0s8  enp0s9  lo
sergey@Server1:~$
```

```
sergey@Server1:/sys/class/net/enp0s3$ ls
addr_assign_type    dev_port            name_assign_type      speed
address             dormant             napi_defer_hard_irqs  statistics
addr_len            duplex              netdev_group          subsystem
broadcast           flags               operstate             testing
carrier             gro_flush_timeout   phys_port_id          threaded
carrier_changes     ifalias             phys_port_name        tx_queue_len
carrier_down_count  ifindex             phys_switch_id        type
carrier_up_count    iflink              power                 uevent
device              link_mode           proto_down
dev_id              mtu                 queues
sergey@Server1:/sys/class/net/enp0s3$ cat operstate
up
sergey@Server1:/sys/class/net/enp0s3$ cat mtu
1500
sergey@Server1:/sys/class/net/enp0s3$ cat address
08:00:27:ce:38:02
```

Identify Ethernet Interfaces

$ ifconfig (legacy)

```
sergey@Server1:~$ ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.105  netmask 255.255.255.0  broadcast 192.168.1.
255
        inet6 fe80::a00:27ff:fe7f:4c00  prefixlen 64  scopeid 0x20<link
>
        ether 08:00:27:7f:4c:00  txqueuelen 1000  (Ethernet)
        RX packets 1560  bytes 595144 (595.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 663  bytes 80251 (80.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

$ ip addr

```
sergey@Server1:~$ ip addr sh dev enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel st
ate UP group default qlen 1000
    link/ether 08:00:27:7f:4c:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.105/24 brd 192.168.1.255 scope global noprefixroute
enp0s3
       valid_lft forever preferred_lft forever
    inet 192.168.1.200/24 scope global secondary enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe7f:4c00/64 scope link
       valid_lft forever preferred_lft forever
```

Identify Ethernet Interfaces

Another application that can help identify all network interfaces available to your system is the *lshw* command. It provides **greater details** around the hardware capabilities of specific adapters: bus information, driver details and all supported

```
sergey@Server1:~$ sudo lshw -class network
  *-network:0
       description: Ethernet interface
       product: 82540EM Gigabit Ethernet Controller
       vendor: Intel Corporation
       physical id: 3
       bus info: pci@0000:00:03.0
       logical name: enp0s3
       version: 02
       serial: 08:00:27:7f:4c:00
       size: 1Gbit/s
       capacity: 1Gbit/s
       width: 32 bits
       clock: 66MHz
       capabilities: pm pcix bus_master cap_list ethernet physical tp 1
0bt 10bt-fd 100bt 100bt-fd 1000bt-fd autonegotiation
       configuration: autonegotiation=on broadcast=yes driver=e1000 dri
verversion=5.11.0-44-generic duplex=full ip=192.168.1.105 latency=64 li
nk=yes mingnt=255 multicast=yes port=twisted pair speed=1Gbit/s
       resources: irq:19 memory:f0200000-f021ffff ioport:d020(size=8)
```

$ sudo lshw -class network

Ethernet Interface Settings

- *ethtool* is a program that **displays** and **changes** Ethernet card settings such as auto-negotiation, port speed, duplex mode, and Wake-on-LAN.
- To view *ethtool* options: *$ethtool -h*

```
sergey@Server1:~$ sudo ethtool enp0s3
Settings for enp0s3:
        Supported ports: [ TP ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Supported pause frame use: No
        Supports auto-negotiation: Yes
        Supported FEC modes: Not reported
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Speed: 1000Mb/s
        Duplex: Full
        Port: Twisted Pair
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: on
        MDI-X: off (auto)
        Supports Wake-on: umbg
        Wake-on: d
        Current message level: 0x00000007 (7)
                               drv probe link
        Link detected: yes
```

Ethernet Interface Logical Names

- Interface logical names can also be configured via a *netplan* configuration.
- If you would like control which interface receives a particular logical name use the *match* and *set-name* keys.
- The *match* key is used to find an adapter based on some criteria like MAC address, driver, etc.
- Then the *set-name* key can be used to change the device to the desired logial name.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth_lan0:
      dhcp4: true
      match:
        macaddress: 00:11:22:33:44:55
      set-name: eth_lan0
```

Hostname configuration

- You can use the *hostname* command or *hostnamectl* command to see or set the system's host name.
- The host name or computer name is usually at system startup in /etc/hostname file.

```
sergey@Server1:/etc/network$ hostname
Server1
sergey@Server1:/etc/network$ hostnamectl
   Static hostname: Server1
   Pretty hostname: Client_2
         Icon name: computer-vm
           Chassis: vm
        Machine ID: c47d7d6709164db09d4a80d265cd1f3f
           Boot ID: 990649f895ba4db68a85364c85909ca4
    Virtualization: oracle
  Operating System: Ubuntu 20.04.3 LTS
            Kernel: Linux 5.11.0-44-generic
      Architecture: x86-64
```

```
sergey@Server1:/etc/network$ sudo hostname Server2
sergey@Server1:/etc/network$ hostname
Server2
sergey@Server1:/etc/network$
```

Linux IP address config

IP Addressing

- Temporary IP Address Assignment
- Permanent IP Address Assignment
- Dynamic IP Address Assignment (DHCP Client)

Temporary IP configuration

- For **temporary** network configurations, you can use the ***ip*** command which is also found on most other GNU/Linux operating systems.
- The ***ip*** command allows you to configure settings which take effect **immediately**, however they are not persistent and **will be lost after a reboot**.
- Syntax and ***ip*** utility options:

ip [options] object command [parameters]

  - **Options** are global settings that affect the operation of the entire utility regardless of other arguments, they do not need to be specified, the option is preceded by a sign «-».
  - **Object** is a data type that you will need to work with, for example: addresses, devices, arp table, routing table, and so on.
  - **Commands** - any action with an object.
  - **Parameters** - commands sometimes need to pass parameters, they are passed at this point.

ip [options] object command [parameters]

- **The most popular options**:

-s, -stats - turns on the output of statistical information.
-d, -details - show even more details.
-o, -oneline - output each record on a new line.
-br, -brief - only display basic information for readability.
-4 - display information for ipv4.
-6 - display information for ipv6.

- **Some Objects:**

address or a - network addresses.
link or l - physical network device.
neighbor or neigh - view and control ARP.
route or r - control routing.
tunnel or t - configure tunneling.

- **Basic commands**:

add,
change,
del or delete,
flush,
get,
list or show,
monitor,
replace,
restore,
save,
set,
show
update.

If no command is given, the default is **show**.

Some IP commands samples

- Sample: *sudo ip addr add 192.168.1.200/24 dev enp0s3*
- The *ip* can then be used to set the link **up** or **down**.
- Sample: *ip link set dev enp0s3 up*

  ip link set dev enp0s3 down
- To verify the IP address configuration:

  ip address show dev enp0s3

```
sergey@Server1:~$ ip addr sh dev enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel st
ate UP group default qlen 1000
    link/ether 08:00:27:7f:4c:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.105/24 brd 192.168.1.255 scope global noprefixroute
enp0s3
       valid_lft forever preferred_lft forever
    inet 192.168.1.200/24 scope global secondary enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe7f:4c00/64 scope link
       valid_lft forever preferred_lft forever
```

Temporary default gateway configuration

- To configure a default gateway: *sudo ip route add default via 10.1.1.1*
- To verify your default gateway configuration: *ip route show*
- Possible variants of *ip route show* output:

  Temporary - default via 10.1.1.1 dev enp0s3

  DHCP - default via 10.1.1.1 dev enp0s3 proto dhcp src 10.1.1.2 metric 100

  Static - default via 10.1.1.1 dev enp0s3 proto static
- If you no longer need this configuration and wish **to purge all IP configuration** from an interface*: ip addr flush enp0s3*

Using *ifconfig* for ip address configuration (legacy)

IP-address configuration:

$sudo ifconfig enp0s3 192.168.0.110 netmask 255.255.255.0

Default gateway configuration:

$ sudo route add default gw 192.168.0.1 enp0s3

Permanent IP Address Assignment

- To make permanent IP address assignment, or others network configuration options, it is needed to edit or, sometimes, to create special file or files:

  /etc/network/interfaces (Ubuntu before 18.04)

  /etc/netplan/*.yaml (Ubuntu starting with 18.04)

  /etc/sysconfig/network-scripts/ifcfg-[network_device_name] (CentOS, Red Hat)

- Once you make changes in the server network configuration file, then require to restart the server networking service in order to reflect the changes:
  - *# sudo systemctl restart networking* - Ubuntu before 18.04, CentOS, Red Hat
  - *# netplan apply* - Ubuntu starting with 18.04

Netplan (Ububtu)

- Starting with Ubuntu 18.04, the default configuration utility is **Netplan**.
- All network configuration files located in **/etc/netplan**. File name may be any, but file extension is mandatory **.yaml**. If there are several files, then they are processed in alphabetical order.
- The essential difference between Netplan and previous Ubuntu versions configuration is that the configuration is written in **YAML**. The main feature of this language is that it is **very sensitive** to spaces. **Don’t use Tab key!!!!** Only 2 or 4 spaces
- Netplan provides the ability to check network settings before applying them: *netplan try*
- The configuration can then be applied using the netplan command: *sudo netplan apply*

Static or dynamic (DHCP client) IP Address Assignment

- To configure your system to use static or dynamic (DHCP client) address assignment, create a netplan configuration in the file /etc/netplan/*.yaml
- Renderer is a network manager: networkd (by default) or NetworkManager
- The addresses field implies a list of values that are enclosed in square brackets and separated by commas, the brackets are required, even if the value is one.
- IP address notation format: address / prefix
- To enable DHCP over IPv4, use the dhcp4 option, in which you can specify both ***true***/***false*** and ***yes***/***no***

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      addresses: [192.168.1.105/24]
      gateway4: 192.168.1.1
      dhcp4: no
    enp0s8:
      addresses: [10.0.1.1/24]
    enp0s9:
      addresses: [10.0.3.1/24]
```

```
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
#     addresses: [10.0.4.2/24]
      dhcp4: true
    enp0s8:
      addresses: [10.0.5.2/24]
```

Static IP Address Assignment (continuation)

```
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      addresses: [192.168.0.200/24, 10.10.10.1/24, 1::1/64]
      gateway4: 192.168.0.1
      dhcp4: no
      nameservers:
```

```
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
#     addresses: [10.0.4.2/24]
      dhcp4: true
    enp0s8:
      addresses:
        - 10.0.5.2/24
        - 10.10.10.1/24
        - 1::1/64
```

```
sergey@Server1:~$ ip addr sh dev enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel st
ate UP group default qlen 1000
    link/ether 08:00:27:ce:38:02 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.200/24 brd 192.168.0.255 scope global noprefixroute
enp0s3
       valid_lft forever preferred_lft forever
    inet 10.10.10.1/24 brd 10.10.10.255 scope global noprefixroute enp0
s3
       valid_lft forever preferred_lft forever
    inet6 1::1/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fece:3802/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Loopback Interface

- The loopback interface is **identified by the system** as *lo* and has a default IP address of 127.0.0.1. It can be viewed using the *ip* command.

```
sergey@Server1:~$ ip addr sh lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN gro
up default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
```

Red Hat, CentOS IP configuration

Network configuration in CentOS

- To identify all available Ethernet interfaces, you can use the commands:

  $ ifconfig

  $ ip addr

  $ nmcli d

- Network's configuration files are in /etc/sysconfig/network-scripts/ . Files names are: ifcfg-[network_device_name], for example: ifcfg-enp0s3

```
[osboxes@osboxes network-scripts]$ nmcli d
DEVICE  TYPE      STATE         CONNECTION
enp0s3  ethernet  connected     enp0s3
enp0s8  ethernet  connected     test
enp0s9  ethernet  disconnected  --
```

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
NAME=enp0s3
UUID=38850678-8c5d-47fb-894e-368bd2056ba6
DEVICE=enp0s3
ONBOOT=yes
```

Network configuration files *ifcfg-<interface-name>*

- Supply the configuration information for each network interface.
- Contain a series of keywords and values parsed at boot time.
- Values are used to configure network interfaces, net masks, and host names; set default gateways; and perform other tasks required to bring the host up on the network.
- The following is a sample ifcfg-eth0 file for a system using static IP address:

```
DEVICE=“eth0”                    # The interface name
BOOTPROTO=“none”                 # Set to “dhcp” to use DHCP
IPADDR=“172.16.205.99”           # Host’s static IP address
NETWORK=“172.16.205.96”          # The network number
BROADCAST=”172.16.205.127”  # The broadcast address
NETMASK=“255.255.255.240”    # The netmask
ONBOOT=“yes”                     # yes to configure at boot
USERCTL=“no”                     # Non-root user are not allowed to control this device
```

Static IP configuration in CentOS

- For CentOS 8 install network configuration scripts:

  $ yum install network-scripts

- To set a static IP for your network, you need to change the BOOTPROTO line to have the value “none".
- Then, add the following information about your network under the already existing text:

  IPADDR0=...

  PREFIX0=...

  GATEWAY0=...

  DNS1=...

- For these changes to take effect, you must restart the network with the command:

  $ systemctl restart network

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
NAME=enp0s3
UUID=38850678-8c5d-47fb-894e-368bd2056ba6
DEVICE=enp0s3
ONBOOT=yes
IPADDR0=10.0.1.40
PREFIX0=24
GATEWAY0=10.0.1.1
DNS1=8.8.8.8
```

Using *nmcli* for IP configuration

- The nmcli (NetworkManager Command Line Interface) command-line utility is used for controlling NetworkManager and reporting network status.
- The nmcli utility can be used by both users and scripts for controlling NetworkManager:
  - For servers, headless machines, and terminals, nmcli can be used to control NetworkManager directly, without GUI, including creating, editing, starting and stopping network connections and viewing network status.
  - For scripts, nmcli supports a terse output format which is better suited for script processing. It is a way to integrate network configuration instead of managing network connections manually.

Nmcli basics

- The basic format of a nmcli command is as follows:

  nmcli [OPTIONS] OBJECT { COMMAND | help } *nmcli con down id enpos8*

where OBJECT can be one of the following options: general, networking, radio, connection, device, agent, and monitor.

- Some of useful optional OPTIONS to get started are:

  nmcli –t or *nmcli –p* (-t, terse; -p, pretty) nmcli con down (up)

```
[osboxes@osboxes ~]$ nmcli -t device
enp0s3:ethernet:connected:enp0s3
enp0s8:ethernet:connected:enp0s8
lo:loopback:unmanaged:
[osboxes@osboxes ~]$ nmcli -p device
====================
  Status of devices
====================
DEVICE  TYPE      STATE      CONNECTION
-----------------------------------------
enp0s3  ethernet  connected  enp0s3
enp0s8  ethernet  connected  enp0s8
lo      loopback  unmanaged  --
```

```
[osboxes@osboxes ~]$ nmcli con down id enp0s8
Connection 'enp0s8' successfully deactivated (D-Bus active path
iveConnection/4)
[osboxes@osboxes ~]$ nmcli con show
NAME    UUID                                  TYPE      DEVICE
enp0s3  38850678-8c5d-47fb-894e-368bd2056ba6  ethernet  enp0s3
enp0s8  8ffba605-0728-4036-be18-a3d1cb609773  ethernet  --
[osboxes@osboxes ~]$ nmcli con up id enp0s8
Connection successfully activated (D-Bus active path: /org/free
on/5)
[osboxes@osboxes ~]$ nmcli con show
NAME    UUID                                  TYPE      DEVICE
enp0s3  38850678-8c5d-47fb-894e-368bd2056ba6  ethernet  enp0s3
enp0s8  8ffba605-0728-4036-be18-a3d1cb609773  ethernet  enp0s8
[osboxes@osboxes ~]$
```

Connection creation

nmcli connection add type ethernet con-name <con-name> ifname <if-name>

```
NAME    UUID                                  TYPE      DEVICE
enp0s3  38850678-8c5d-47fb-894e-368bd2056ba6  ethernet  enp0s3
test    57ddc5bb-cd00-440a-aab1-90bec0585a27  ethernet  --
[osboxes@osboxes network-scripts]$ nmcli conn del test
Connection 'test' (57ddc5bb-cd00-440a-aab1-90bec0585a27) successfully deleted.
[osboxes@osboxes network-scripts]$ nmcli con sh
NAME    UUID                                  TYPE      DEVICE
enp0s3  38850678-8c5d-47fb-894e-368bd2056ba6  ethernet  enp0s3
[osboxes@osboxes network-scripts]$ nmcli con add ifname enp0s8 type ethernet con-name "test"
Connection 'test' (b82ecbf5-abf2-4b7e-8aff-2f500a734c29) successfully added.
[osboxes@osboxes network-scripts]$ nmcli con mod test +ipv4.addr "10.0.5.5/24"
[osboxes@osboxes network-scripts]$ nmcli con up test
^CError: nmcli terminated by signal Interrupt (2)
[osboxes@osboxes network-scripts]$ nmcli con sh
NAME    UUID                                  TYPE      DEVICE
test    b82ecbf5-abf2-4b7e-8aff-2f500a734c29  ethernet  enp0s8
enp0s3  38850678-8c5d-47fb-894e-368bd2056ba6  ethernet  enp0s3
[osboxes@osboxes network-scripts]$ ping 10.0.5.2
PING 10.0.5.2 (10.0.5.2) 56(84) bytes of data.
64 bytes from 10.0.5.2: icmp_seq=1 ttl=64 time=0.916 ms
64 bytes from 10.0.5.2: icmp_seq=2 ttl=64 time=1.14 ms
64 bytes from 10.0.5.2: icmp_seq=3 ttl=64 time=1.01 ms
```

Adding (deleting) ip addres

1. nmcli con modify <con-name> +(-) ipv4.addr “<ip_addr/pref>”
2. nmcli con up <con-name>

```
[osboxes@osboxes ~]$ ip addr show dev enp0s8
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:28:d9:f5 brd ff:ff:ff:ff:ff:ff
    inet 10.0.5.10/24 brd 10.0.5.255 scope global noprefixroute enp0s8
       valid_lft forever preferred_lft forever
[osboxes@osboxes ~]$
[osboxes@osboxes ~]$ nmcli con mod enp0s8 +ipv4.addr "10.0.5.20/24"
[osboxes@osboxes ~]$
[osboxes@osboxes ~]$ nmcli con up enp0s8
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnecti
on/10)
[osboxes@osboxes ~]$
[osboxes@osboxes ~]$ ip addr show dev enp0s8
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:28:d9:f5 brd ff:ff:ff:ff:ff:ff
    inet 10.0.5.10/24 brd 10.0.5.255 scope global noprefixroute enp0s8
       valid_lft forever preferred_lft forever
    inet 10.0.5.20/24 brd 10.0.5.255 scope global secondary noprefixroute enp0s8
       valid_lft forever preferred_lft forever
[osboxes@osboxes ~]$
```

DNS client configuration

Name Resolution (Ubuntu)

- Name resolution as it relates to IP networking is the process of mapping IP addresses to hostnames
- **systemd-resolved** is a system service that provides network name resolution to local applications. It implements a caching and validating DNS/DNSSEC stub resolver.
- Traditionally, the file */etc/resolv.conf* is a static configuration file that rarely needed to be changed or automatically changed via DCHP client hooks. It contains default entry: nameserver **127.0.0.53**
- File */run/systemd/resolve/resolv.conf* generates dynamically and includes the actual DNS-server address. This file may be symlinked from /etc/resolv.conf

```
sergey@Server1:~$ more /etc/resolv.conf
nameserver 127.0.0.53
options edns0 trust-ad
sergey@Server1:~$
```

```
sergey@Server1:~$ nslookup
> epam.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   epam.com
Address: 3.214.134.159
>
```

Name Resolution Configuration (Ubuntu)

- To configure the resolver, add the IP addresses of the nameservers that are appropriate for your network to the Netplan configuration file:

```
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      addresses: [192.168.1.105/24]
      gateway4: 192.168.1.1
      dhcp4: no
      nameservers:
        addresses: [8.8.8.8, 8.8.8.4]
```

```
sergey@Server1:~$ more /run/systemd/resolve/resolv.conf
# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients direc
tly to
# all known uplink DNS servers. This file lists all configured search d
omains.
#
# Third party programs must not access this file directly, but only thr
ough the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a differ
ent way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported m
odes of
# operation for /etc/resolv.conf.

nameserver 8.8.8.8
nameserver 8.8.8.4
```

systemd-resolve

- To check the actual name servers, use the command: *$systemd-resolve --status*
- For manual name resolving may be used command: *$systemd-resolve <server-name>*
- For setting DNS server address may be used command: *$systemd-resolve --set-dns=SERVER*
- For flushing DNS cash may be used command: *$systemd-resolve --flush-caches*

```
Link 2 (enp0s3)
      Current Scopes: DNS
DefaultRoute setting: yes
       LLMNR setting: yes
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
  Current DNS Server: 8.8.8.8
         DNS Servers: 8.8.8.8
                      8.8.8.4
          DNS Domain: ~.
```

```
sergey@Server1:~$ systemd-resolve epam.com
epam.com: 3.214.134.159
```

Name Resolution *nslookup*

- **nslookup cisco.com** – request for default dns server for resolving domain name (cisco.com)
- **nslookup cisco.com 8.8.8.8** – request for explicit pointed dns server (8.8.8.8) for resolving domain name (cisco.com)
- **nslookup -type=mx cisco.com** – request ip address of the domain (cisco.com) mail servers
- **nslookup -type=any cisco.com 8.8.8.8** - request of any ip address for domain (cisco.com)
- **nslookup 91.206.200.104** - reverse lookup

```
sergey@Server1:/etc$ nslookup -type=ns cisco.com 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
cisco.com       nameserver = ns3.cisco.com.
cisco.com       nameserver = ns1.cisco.com.
cisco.com       nameserver = ns2.cisco.com.

Authoritative answers can be found from:

sergey@Server1:/etc$ nslookup cisco.com ns2.cisco.com
Server:         ns2.cisco.com
Address:        64.102.255.44#53

Name:   cisco.com
Address: 72.163.4.185
Name:   cisco.com
Address: 2001:420:1101:1::185

sergey@Server1:/etc$ nslookup 72.163.4.185
185.4.163.72.in-addr.arpa       name = redirect-ns.cisco.com.

Authoritative answers can be found from:
```

Name Resolution *Static* Hostnames

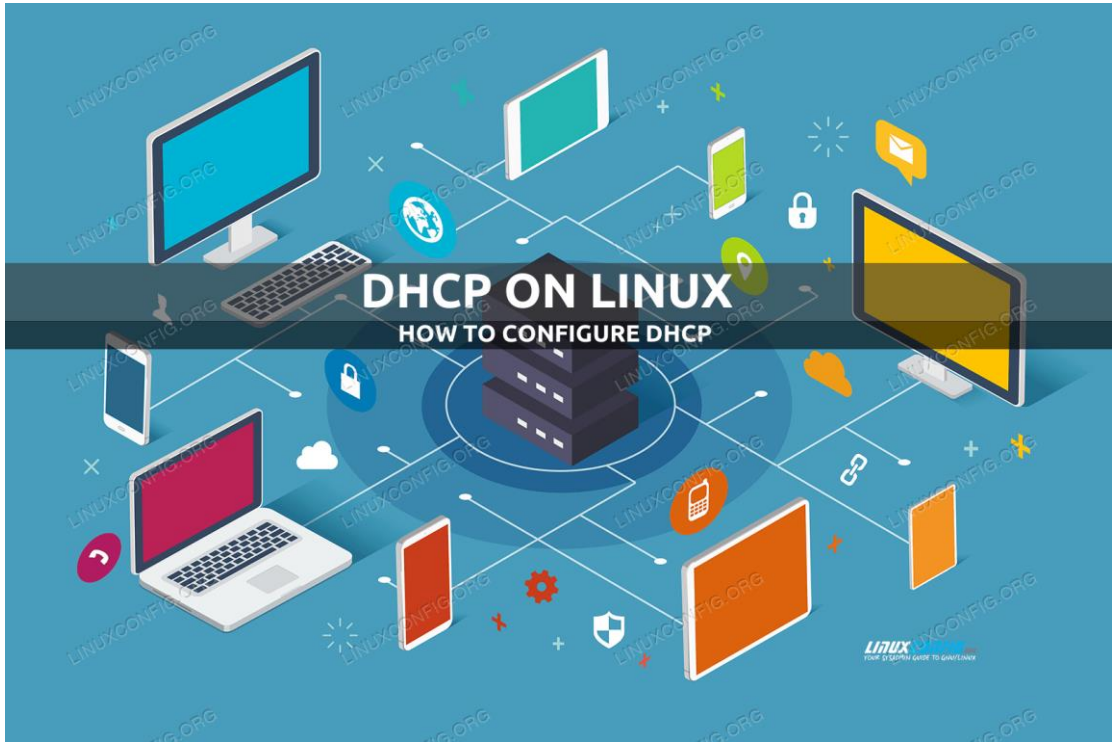
- Static hostnames are locally defined hostname-to-IP mappings located in the file **/etc/hosts**.
- Entries in the hosts file will have precedence over DNS **by default**. This means that if your system tries to resolve a hostname and it matches an entry in /etc/hosts, it **will not attempt** to look up the record in DNS.
- In some configurations, especially when Internet access is not required, servers that communicate with a limited number of resources can be conveniently set to use static hostnames instead of DNS. Example:

```
127.0.0.1   localhost
10.0.0.11   server1 server1.example.com vpn
10.0.0.12   server2 server2.example.com mail
10.0.0.13   server3 server3.example.com www
```

DHCP server install and config

DHCP Introduction

- DHCP service enables devices on a network to obtain IP addresses, subnet masks, gateway, and other IP networking parameters **dynamically** from a DHCP server.
- DHCP server is contacted, and address requested - chooses address from a configured range of addresses called a **pool** and “leases” it to the host for a **set period**
- DHCP used for general purpose hosts such as **end user devices.**

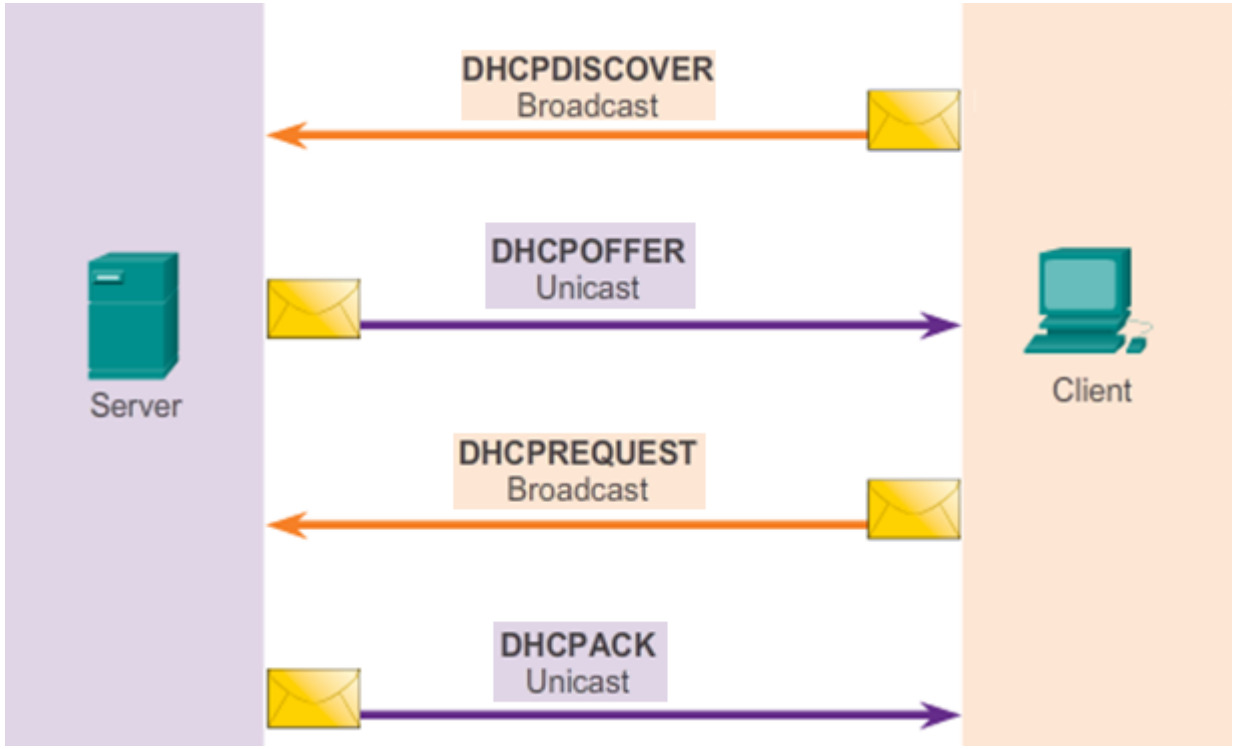


DHCP Operation

DHCPv4 uses three different address allocation methods:

- **Manual Allocation (MAC address)** - This ensures that a particular address is assigned automatically to network card, based on it’s MAC address.
- **Automatic Allocation** - DHCPv4 automatically assigns a static IPv4 address permanently to a device, selecting it from a pool of available addresses. No lease.
- **Dynamic Allocation** - DHCPv4 dynamically assigns, or leases, an IPv4 address from a pool of addresses for a limited period of time chosen by the server, or until the client no longer needs the address. Most commonly used.

DHCPv4 Lease Origination



Ubuntu DHCP server installation

- The DHCP server Ubuntu makes available is **dhcpd** (dynamic host configuration protocol daemon), which is easy to install and configure and will be automatically started at system boot.
- At a terminal prompt, enter the following command to install dhcpd:

  $ sudo apt install isc-dhcp-server

- dhcpd's messages are being sent to syslog: /var/log/syslog
- To change the default configuration, edit /etc/dhcp/dhcpd.conf to suit your needs and particular configuration.
- After changing the config files, you must **restart** the dhcpd service:

  $ sudo systemctl restart isc-dhcp-server.service

dhcpd.conf overview

- The dhcpd.conf file essentially consists of a list of statements. Statements fall into two broad categories - **parameters** and **declarations**.
- **Parameter** statements either say how to do something (e.g., how long a lease to offer), whether to do something (e.g., should dhcpd provide addresses to unknown clients), or what parameters to provide to the client (e.g., use gateway 10.0.1.1).
- **Declarations** are used to describe clients on the network, to provide addresses that can be assigned to clients etc. The most popular declarations are: **subnet**, **host**, **group, pool**

Ubuntu DHCP server configuration

```
default-lease-time 600;
max-lease-time 7200;
authoritative;
subnet 192.168.1.0 netmask 255.255.255.0
{
  range 192.168.1.150 192.168.1.200;
  option routers 192.168.1.254;
  option domain-name-servers 192.168.1.1, 192.168.1.2;
  option domain-name "mydomain.example";
}
host Client1
{
  hardware ethernet 00:02:3f:3d:73:b3;
  fixed-address 192.168.1.100;
}
```

- The **max-lease-time** is the maximum lease time that you will get. If you ask for (say) 10000 seconds and the maximum lease time is (say) 7200 seconds, then you will get a lease time of 7200 seconds.
- The **default lease time** is the lease time you will get if you don't request any particular lease time.
- In default configuration uncomment the line **#authoritative**;

Ubuntu DHCP server configuration

```
default-lease-time 600;
max-lease-time 7200;
authoritative;
subnet 192.168.1.0 netmask 255.255.255.0
{
  range 192.168.1.150 192.168.1.200;
  option routers 192.168.1.254;
  option domain-name-servers 192.168.1.1, 192.168.1.2;
  option domain-name "mydomain.example";
}
host Client1
{
  hardware ethernet 00:02:3f:3d:73:b3;
  fixed-address 192.168.1.100;
}
```

- **Subnet** <*Subnet_addr*> **netmask** <*Net_mask*> describes the subnet from which the ip addresses will be leased.
- The **range** <*first ip-addr*> <*last ip-addr*> defines a range of available addresses. Addresses are allocated sequentially from first to last.
- **option routers** <*ip-addr*> defines a default-gateway address for clients
- **option domain-name-servers** <*ip-addr*> defines a DNS server address for clients
- **option domain-name “**<*domain-name*>**“** defines a domain name for client’s hosts

Ubuntu DHCP server configuration

```
default-lease-time 600;
max-lease-time 7200;
authoritative;
subnet 192.168.1.0 netmask 255.255.255.0
{
  range 192.168.1.150 192.168.1.200;
  option routers 192.168.1.254;
  option domain-name-servers 192.168.1.1, 192.168.1.2;
  option domain-name "mydomain.example";
}
host Client1
{
  hardware ethernet 00:02:3f:3d:73:b3;
  fixed-address 192.168.1.100;
}
```

- **Host** <*Name*> - setting a predefined IP address for a specific host
- **hardware ethernet** <*MAC-address*> - host's MAC-address.
- **fixed-address** <*ip-addr*> a predefined IP address

Sometimes needed to edit **/etc/default/isc-dhcp-server** to specify the interfaces dhcpd should listen to: INTERFACESv4="enp0s3"

Some other useful declarations and statements

- The **pool** declaration can be used to specify a pool of addresses that will be treated differently than another pool of addresses, even on the same network segment or subnet.
- The **group** declaration can be used If parameters are to be applied to a group of declarations which are not related strictly on a per-subnet basis.
- The **allow** and **deny** statements can be used to control the response of the DHCP server to various sorts of requests. The **unknown-clients** flag is used to tell dhcpd whether or not to dynamically assign addresses to unknown clients. An unknown client is a client that has no **host** declaration.

DHCP client configuration

- To configure your **Ubuntu** system to use dynamic (DHCP client) address assignment, create a netplan configuration in the file /etc/netplan/*.yaml
- To enable DHCP over IPv4, use the dhcp4 option, in which you can specify both ***true***/***false*** and ***yes***/***no***

```
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
#     addresses: [10.0.4.2/24]
      dhcp4: true
    enp0s8:
      addresses: [10.0.5.2/24]
```

- Network's configuration files **CentOS** system are in /etc/sysconfig/network-scripts/.
- Files names are: ifcfg-[network_device_name], for example: ifcfg-enp0s3

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
NAME=enp0s3
```

Ubuntu DHCP server verification

- To review all actual leased IP-addresses:

$ dhcp-lease-list

```
sergey@Server1:~$ dhcp-lease-list
To get manufacturer names please download http://standards.ieee.org/reg
auth/oui/oui.txt to /usr/local/etc/oui.txt
Reading leases from /var/lib/dhcp/dhcpd.leases
MAC                IP              hostname       valid until         m
anufacturer
==========================================================================
========================
08:00:27:9d:ad:3b  10.0.3.10       Client2        2022-01-27 12:37:48 -
NA-
08:00:27:9e:e9:1a  10.0.1.12       Client1        2022-01-27 12:34:48 -
NA-
```

- The history of leased IP-addresses is in /var/lib/dhcp/dhcpd.leases

```
lease 10.0.1.12 {
  starts 4 2022/01/27 12:14:48;
  ends 4 2022/01/27 12:24:48;
  cltt 4 2022/01/27 12:14:48;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 08:00:27:9e:e9:1a;
  uid "\001\010\000'\236\351\032";
  client-hostname "Client1";
}
lease 10.0.3.10 {
  starts 4 2022/01/27 12:17:48;
  ends 4 2022/01/27 12:27:48;
  cltt 4 2022/01/27 12:17:48;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 08:00:27:9d:ad:3b;
  uid "\001\010\000'\235\255;";
  client-hostname "Client2";
```

```
lease 10.0.3.11 {
  starts 3 2022/01/26 18:23:36;
  ends 3 2022/01/26 18:33:36;
  tstp 3 2022/01/26 18:33:36;
  cltt 3 2022/01/26 18:23:36;
  binding state free;
  hardware ethernet 08:00:27:9e:e9:1a;
  uid "\001\010\000'\236\351\032";
```

Useful links

- Ubuntu network configuration - https://ubuntu.com/server/docs/network-configuration
- CONFIGURING IP NETWORKING WITH NMCLI (Red Hat, CentOS) - https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/networking_guide/sec-configuring_ip_networking_with_nmcli

Q&A


Thank you!