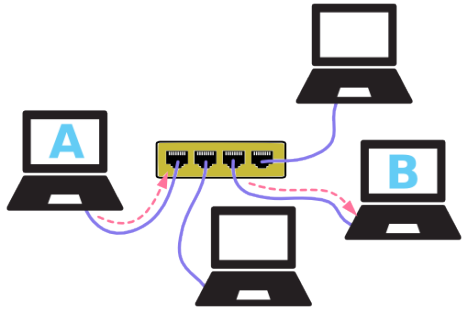

= Session-(1.6) =

Linux Networking Configuration

Considering Ubuntu/Debian Based Distributions

So far reviews of yesterday's sessions;

=> Introduction of Virtualization.

=> Install The Operating System. We faced some issues;
   → Faulty ISO/boot Image.
   → UEFI BIOS Boot Issue.
   → Configuring The IP address.
   → Default Ubuntu Package Repository Was not working.

=> Learned Some Basic Linux Commands.
    → Editing Files In Terminal / by vim.
=> Networking Configuration and Connecting to Internet.
      → Restore Classic Networking Configuration Option.
=> Selecting Local Ubuntu Repository Mirror.

We can connect a Linux Workstation/Server to the network in several ways. We already saw some configuration method in session (1.4).

It has some more method, also in two major Linux distribution (debian-based & redhat-based) the configuration file location and configuration syntax are different.

Also there is command line method, which is universal to almost all Linux distributions, but it is temporary & it resets after every reboot unless we create some scripts and hook it to Linux-startup system.

So, considering the above aspects, we can list them as follows:-

- → File Based Configuration.
- → Command Line Configuration.
- → Script Based Configuration (Commands in a file).

Also there are other aspects we have to consider, which are listed bellow,

- → Direct Ethernet Connection
- → Bridged Ethernet Connection. (Require for Virtualization)
- → 801.1q VLAN Based Connection.
- → Bonding Ethernet (+) Bridge Connection.
- → Bonding Bridge (+) VLAN Based Connection. and so on combined configuration method considering the demand of the solution.

In previous session (1.4) we already saw some command line method and configuration file based method.

In Ubuntu 18.04, the default configuration tool is **netplan**. It is a file based configuration method which syntax is composed with YAML (Ain't Markup Language - yaml.org). Which lacks some features and we are still not used-to with it.

Although in this workshop we avoided netplan and restored the classic options/tools to configure the network.

Here is some configuration syntax of **netplan:**

file /etc/netplan/01-netcfg.yaml (Standard Configuration) | `50-cloud-init.yaml in Server 18.04`

```
network:
    ethernets:
        enp1s0:
            addresses: [192.168.108.11/24]
            gateway4: 192.168.108.1
            nameservers:
                addresses: [8.8.8.8,9.9.9.9]
                search: []
            optional: true
        enp2s0:
            addresses: [172.16.208.7/24]
            optional: true
    version: 2
```

```
sudo netplan try          ; validate your configuration before applying it,
sudo netplan apply        ; apply new settings
```

Here is some configuration syntax of **ifconfig/ifupdown:**

config file: /etc/network/interfaces

```
auto lo
iface lo inet loopback

auto eth1
iface eth1 inet dhcp


auto eth0
iface eth0 inet static
     address 192.168.108.11
     network 192.168.108.0
     netmask 255.255.255.0
     gateway 192.168.108.1
```

Here is some configuration syntax of **ip:**

This is the default ip configuration command line tools now in almost all Linux distribution. We can do even complex network configuration with this tool.

```
ip addr

ip addr add 192.168.108.11/24 dev enp1s0
ip link set enp1s0 up
ip addr show

ip route show
ip route add default via 192.168.108.1
ip route add 192.168.208.0/24 via 192.168.108.8 dev enp1s0


ip addr del 192.168.108.11/24 dev enp1s0
ip route del 192.168.208.0/24


networkctl                          ; to see interfaces
networkctl status                   ; to see interface status
```

Combining these commands, we can create shell-script and hook-it with linux-startup script/system to achieve a persistent network configuration technique.

Some network connectivity troubleshooting tools;

```
arp -n                      ; to see the neighboring devices of same subnet

ping: syntax
ping 192.168.108.11
ping -c 10 -i 0.1 192.168.108.1   (10 time ping, 0.1sec per ping)


arping: syntax (useful if firewall is on in next destination hop)

apt install mtr arping
arping -I enp1s0 192.168.108.1

mtr 192.168.108.1           ; to find the route path and latency in each hop
```

In visualization with Linux, KVM & LXC/LXD OpenVswitch is widely used and perform better than default linux-bridge.

In some next-session we will know more about OpenVswitch and its use.