**-->Linux :**

Just like Windows, iOS, and Mac OS, Linux is an operating system. In fact, one of the most popular platforms on the planet, Android, is powered by

the Linux operating system. An operating system is software that manages all of the hardware resources associated with your desktop or laptop.

To put it simply, the operating system manages the communication between your software and your hardware. Without the operating system (OS), the

software wouldn’t function.

Linux is the best-known and most-used open source operating system.Linux is an open-source Unix-like operating system-based family on the Linux kernel,

and the OS kernel was first published on 17 September 1991 by Linus Torvalds.

***OPEN SOURCE:***

The term open source refers to something people can modify and share because its design is publicly accessible.

***UNIX :***

Unix (trademarked as UNIX) is an operating system (OS) that is one of the first OS developed using the C programming language. It is a multitasking, multi-user OS that provides a set of programs to act as a link between the computer and the user. The primary

design philosophy of Unix is to offer simple yet powerful tools to perform complex tasks. It features a command-line interface allowing

users to communicate with the system using commands than a Graphical User Interface (GUI).

***UNIX-Like :***

An operating system is said to be Unix-based or Unix-like if it's designed to function and behave similar to the Unix operating system.

Unix-like operating systems are not necessarily derived from the original Unix operating system but mimic its behavior. The Unix-Like

operating system includes some improvements that are available in the open source software (OSS). It shares many design principles and

features of the original Unix operating system but not its proprietary code or specific implementations completely.

*Before entering the Linux distribution we have to know some topics:*

Linux distributions, often referred to as "distros," are variations of the Linux operating system that package together different components such as the Linux kernel, system utilities, libraries, and software applications to provide a complete computing environment.

***Kernel:***

The Linux kernel is the core of the operating system, providing essential functionalities such as process

management, memory management, device drivers, and system calls.

## ***Package Management System:***

Most Linux distributions come with a package management system that simplifies the installation, removal, and updating of software packages.

***System Libraries and Utilities***:

Linux distributions include a set of system libraries and utilities that provide essential functionalities

and tools for managing the system, interacting with hardware, and performing various tasks.

## ***Applications:***

Linux distributions include a variety of pre-installed applications, such as web browsers, office suites, multimedia players, and development tools. Users can also install additional applications from software repositories using the package management system.

***Popular Linux Distros:-***

Kali Linux :

* **Purpose:** Kali Linux is designed for digital forensics, penetration testing, and security auditing. It comes pre-installed with numerous penetration testing tools and software suites for conducting security assessments and forensic analysis.
* **Target Audience:** Security professionals, ethical hackers, penetration testers, and individuals interested in cybersecurity.
* **Features:** Kali Linux includes tools for network analysis, vulnerability assessment, web application testing, wireless network assessment, and more

Parrot:

* **Purpose:** Parrot Security OS, often referred to simply as Parrot, is another Debian-based distribution aimed at penetration testing, digital forensics, and privacy protection. It provides a wide array of security-related tools.
* **Target Audience:** Similar to Kali Linux, Parrot is targeted at security professionals, ethical hackers, and individuals interested in cybersecurity.
* **Features:** Parrot includes a variety of tools for penetration testing, vulnerability analysis, cryptography, and anonymity tools such as Tor.

Ubuntu:

* **Purpose:** Ubuntu is one of the most popular and user-friendly Linux distributions. It aims to provide a stable, easy-to-use operating system for desktop, server, and cloud computing.
* **Target Audience:** Ubuntu targets a broad audience, including beginners, developers, businesses, and enthusiasts who seek a reliable and versatile operating system.
* **Features:** Ubuntu comes with a comprehensive set of software applications, including productivity tools, multimedia software, development tools, and more. It emphasizes usability and accessibility.

Fedora:

* **Purpose:** Fedora is a community-driven Linux distribution sponsored by Red Hat. It aims to provide the latest in open-source software while also emphasizing innovation, security, and stability.
* **Target Audience:** Fedora is geared towards developers, enthusiasts, and users who want cutting-edge software and technologies while maintaining stability.
* **Features:** Fedora includes a range of software packages for desktop and server environments. It often serves as a testbed for new features and technologies that may eventually make their way into Red Hat Enterprise Linux (RHEL) and other distributions.

# **Network Configuration using Linux:**

Network configuration is the process of assigning network settings, policies, flows, and controls

In a command-line environment, the commands **ipconfig** (for Windows network configuration) and **ifconfig** (for Linux network configuration, as well as Mac OSX and other Linux-like environments) allow you to view information about your network configuration and to configure your network interface.

 In a [virtual network](https://www.vmware.com/topics/glossary/content/virtual-networking.html), it’s easier to make network configuration changes because physical network devices appliances are replaced by software, removing the need for extensive manual configuration.

1. **Ip :** 
   * ip is a powerful command-line utility for managing network interfaces, IP addresses, routing, and more.
   * With the ip command, you can adjust the way a Linux computer handles IP addresses, network interfaces controllers (NICs), and routing rules.
   * We can use addr , a , address all are equivalent.
   * Examples:
     + *ip addr show*: Displays the IP addresses assigned to all network interfaces.
     + *ip route show*: Shows the routing table.
     + *ip link set <interface> up/down*: Brings a network interface up or down.
     + *ip addr add <ip\_address>/<subnet\_mask> dev <interface>:* Assigns an IP address to a network interface

*(* *sudo ip addr add 192.168.4.44/24 dev enp0s3 )*

* + - *ip -4 addr :* to limit the output to the IP version 4 addresses
    - *ip -6 addr* : to limit the output to the IP version 6 addresses
    - *sudo ip addr del 192.168.4.44/24 dev enp0s3 :* To delete an IP address
    - *ip route* : To see the routes defined on your computer

1. **dig (Domain Information Groper):**
   * dig is a command-line tool for querying DNS servers and retrieving DNS-related information such as IP addresses, domain records, and name servers.
   * Example:
     + *dig example.com*: Retrieves DNS information for the domain "example.com".
     + *dig ubuntu.org fedora.org manjaro.com* :We can also use multiple domain name.
     + *dig -x 209.51.188.148* : If you have an IP address and want to know where it goes, you can try a reverse DNS lookup.
2. **nslookup (Name Server Lookup):**
   * nslookup is another command-line tool used for querying DNS servers to obtain domain-related information.
   * Example:
     + *nslookup example.com*: Performs a DNS lookup for the domain "example.com".
     + *nslookup [IP Address] :* reverse DNS look-up by providing the IP Address

*(nslookup 192.168.0.10)*

1. **netstat:**
   * netstat is a command-line utility for displaying network connections, routing tables, interface statistics, and more.
   * The netstat command lets you discover which sockets are connected and which sockets are listening. Meaning, it tells you which ports are in use and which processes are using them. It can show you routing tables and statistics about your network interfaces and multicast connections.
   * Examples:
     + *netstat -a | less :* The -a (all) option makes netstat show all the connected and waiting sockets. and show in less for pipe less command.
     + *netstat -at | less :* The netstat -a command can provide more information than you need to see. If you only want or need to see the TCP sockets, you can use the -t (TCP) option to restrict the display to *only show TCP sockets.*
     + *netstat -au | less : Here -u (UDP) and -x (UNIX) options behave in a similar way*
     + *netstat -tuln*: Displays listening TCP and UDP connections.
     + *netstat -r*: Shows the routing table.
     + *netstat -l | less* : To see the sockets that are in the listening or waiting state, use the -l (listening) option*.*
2. **nmcli (Network Manager Command-Line Interface):**
   * nmcli is a command-line tool for controlling NetworkManager, which is a service that manages network interfaces and connections in many Linux distributions.
   * Examples:
     + *nmcli connection show*: Lists available network connections.
     + *nmcli device show*: Displays information about network devices.
     + *nmcli general status :*  To check NetworkManager is installed, running, and we can connect to it with nmcli
     + With nmcli you can create a network connection and set some of its configuration options with a single command.
3. **route:**
   * route is a command-line utility for viewing and modifying the IP routing table.
   * Examples:
     + *Route :* To see the current routing table .
     + *route -n*: Shows the routing table in a numeric format.
     + *route add default gw <gateway\_ip>:* Adds a default gateway to the routing table.
     + *route del default gw <gateway\_ip>:* Removes a default gateway from the routing table.

## 

## **Storage Management :-**

1. **Master Boot Record (MBR):**
   * MBR is like a tiny roadmap stored on your computer's hard drive. It tells your computer where to find the operating system so it can start up. It's the first thing your computer checks when you turn it on.
2. **ext3 File System:**
   * Think of the ext3 file system as the way your computer organizes and stores files on its hard drive. It's like a filing cabinet with folders and labels that keeps everything organized and easy to find.
3. **Network File System (NFS):**
   * NFS is like a virtual file-sharing system that allows different computers on a network to access and share files with each other. It's like having a shared folder that everyone can access and contribute to, regardless of where they are on the network.
4. **Samba/SMB:**
   * Samba, also known as SMB (Server Message Block), is like a bridge between different types of operating systems. It allows Windows computers to communicate and share files with Linux or Unix-based systems, making it easier for them to work together and share resources.
5. **New Technology File System (NTFS):**
   * NTFS is a file system primarily used by Windows computers. It's like a more advanced version of the filing system, with features like file encryption and larger storage capacities. It's what Windows uses to organize and store files on its hard drives.

**Linux Boot Process:-**

The boot process is a fundamental aspect of any operating system. It is the process by which the operating system loads into the computer's memory, initializes its components, and prepares to execute user applications

**Bootstrap Phase:**

* BIOS : The BIOS (Basic Input/Output System) is the firmware responsible for initiating the computer's hardware components. The BIOS is the first step in any operating system’s boot process and is independent of the operating system that is to be loaded.
* This is like the first step where the computer starts up.
* When you turn on your computer, it needs to know where to find the operating system. The bootstrap phase helps the computer figure out where the operating system is stored.
* It's like turning on the lights in a dark room so you can find your way around.

**Bootloader Phase:**

* Once the computer knows where the operating system is, it loads a program called the bootloader.
* The bootloader is like a gatekeeper. It checks everything is okay with the operating system before letting it start up.
* It's similar to a bouncer at a club checking IDs before letting people in.

**Kernel Phase:**

* After the bootloader gives the green light, the kernel phase begins.
* After the boot loader loads the kernel into memory, the kernel begins the process of initializing the system. Its primary function is to manage system resources such as memory, CPU, and I/O devices.
* The kernel is like the brain of the operating system. It manages everything the computer does.
* It's like the conductor of an orchestra, directing all the different parts to work together harmoniously.

**Initialization Phase:**

* Once the kernel takes over, it starts setting up all the essential parts of the operating system.
* This includes things like loading device drivers, setting up user accounts, and preparing the system for use

**Cloud and Virtualization :-**

Virtualization is technology that allows you to create multiple simulated environments or dedicated resources from a single, physical hardware system. In virtualization, hypervisor software sits on top of the physical hardware, abstracting and delivering the machine resources to virtual machines.

Cloud computing is a set of principles and approaches to deliver compute, network, and storage infrastructure resources, services, platforms, and applications to users on-demand across any network. Cloud infrastructure is physically off-premises and may include virtualization, or container software you can use to pool and share resources.

**OVF and OVA Templates:**

OVF provides a standard way to describe VM configurations, while OVA is a convenient packaging format that bundles OVF specifications and associated virtual disks into a single file for easy distribution and deployment.

*OVF (Open Virtualization Format):*

* OVF is a standard developed by the Distributed Management Task Force (DMTF) for describing VMs and their configurations in a platform-independent manner.
* It encapsulates a VM, its metadata, and possibly multiple disk images into a single file or directory structure.
* OVF describes the VM's hardware configuration, virtual disk images, networking configurations, and other properties.
* OVF files are typically used for importing and exporting VMs between different virtualization platforms, making them more portable.

*OVA (Open Virtualization Appliance):*

* OVA is a variation of the OVF standard. It packages an OVF VM specification and its associated virtual disks (usually in the VMDK format) into a single file.
* Essentially, OVA is a single-file distribution of an OVF package, making it easier to distribute and deploy VMs.
* OVA files are commonly used for distributing pre-configured VMs, appliances, or software solutions that are ready to run within a virtualized environment.
* By packaging everything into a single file, OVA simplifies the process of sharing and deploying VMs across different virtualization platforms.

To Know briefly : <https://forum.huawei.com/enterprise/en/ovf-vs-ova-vs-vmdk-what-are-they-what-are-the-differences/thread/667285435682013184-667213860102352896>

**Container Technology and Docker Basics:**

* Containers provide a way to package an application and its dependencies into a standardized unit for software development, deployment, and operation.
* They encapsulate the application, runtime environment, libraries, and dependencies into a single package, ensuring consistency across different environments.
* Containers share the host operating system's kernel, making them lightweight and efficient compared to traditional virtual machines.
* Container technology enables faster development cycles, improved scalability, and increased portability of applications

***Why are containers useful?***

* Portability – the isolated environment that containers provide effectively means the container is decoupled from the environment in which they run. Basically, they don’t care much about the environment in which they run, which means they can be run in many different environments with different operating systems and hardware platforms.
* Consistency – since the containers are decoupled from the environment in which they run, you can be sure that they operate the same, regardless of where they are deployed. The isolated environment that they provide is the same across different deployment environments.
* Speed to deploy – for the same reasons as above. There is no need for considerations around how the application will operate in a production environment. If it runs in a container in one environment (say, your local machine), then it can be made to run in a container in another environment (say, in a cloud provider) very quickly

**Docker:**

* Docker is a platform for developing, shipping, and running applications in containers.
* Docker is a set of platforms as a service (PaaS) products that use Operating system-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries, and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating system kernel and therefore use fewer resources than a virtual machine.
* Docker is an open-source containerization platform by which you can pack your application and all its dependencies into a standardized unit called a container. Containers are light in weight which makes them portable and they are isolated from the underlying infrastructure and from each other container. You can run the docker image as a docker container in any machine where docker is installed without depending on the operating system.

It provides tools and APIs for building, distributing, and managing containers across different environments.

Key components of Docker include:

* Docker Engine: The runtime environment for containers, responsible for creating, running, and managing containers.
* Dockerfile: A text file that contains instructions to build a Docker image. It specifies the base image, dependencies, and commands to run within the container.
* Docker Image: A lightweight, standalone, executable package that includes everything needed to run a piece of software, including code, runtime, libraries, and dependencies.
* Docker Container: An instance of a Docker image that runs as a lightweight, isolated process on the host system.
* Docker Hub: A public registry of Docker images where users can share and discover pre-built images for various applications and services.
* Docker Compose: A tool for defining and running multi-container Docker applications. It uses a YAML file to configure the services, networks, and volumes required for the application.
* Docker Swarm and Kubernetes: Tools for orchestrating and managing clusters of Docker containers at scale, providing features like service discovery, load balancing, and automated scaling.

To Know more about container and docker:

* <https://www.geeksforgeeks.org/introduction-to-docker/>
* <https://endjin.com/blog/2022/01/introduction-to-containers-and-docker>