**-->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>

**Cloud:**

The cloud refers to servers that are accessed over the Internet, and the software and databases that run on those servers. Cloud servers are located in data centers all over the world. By using cloud computing, users and companies do not have to manage physical servers themselves or run software applications on their own machine.

The cloud enables users to access the same files and applications from almost any device, because the computing and storage takes place on servers in a data center, instead of locally on the user device.

Cloud computing is possible because of a technology called virtualization. Virtualization allows for the creation of a simulated, digital-only "virtual" computer that behaves as if it were a physical computer with its own hardware.

*Cloud services:*

The resources available in the cloud are known as "services," since they are actively managed by a cloud provider.

**Software-as-a-Service (SaaS)**: Instead of users installing an application on their device, [SaaS](https://www.cloudflare.com/learning/cloud/what-is-saas/) applications are hosted on cloud servers, and users access them over the Internet. SaaS is like renting a house: the landlord maintains the house, but the tenant mostly gets to use it as if they owned it.

**Platform-as-a-Service (PaaS)**: In this model, companies don't pay for hosted applications; instead they pay for the things they need to build their own applications. [PaaS](https://www.cloudflare.com/learning/serverless/glossary/platform-as-a-service-paas/) vendors offer everything necessary for building an application, including development tools, infrastructure, and operating systems, over the Internet. PaaS can be compared to renting all the tools and equipment necessary for building a house, instead of renting the house itself.

**Infrastructure-as-a-Service (IaaS)**: In this model, a company rents the servers and storage they need from a cloud provider. They then use that cloud infrastructure to build their applications. [IaaS](https://www.cloudflare.com/learning/cloud/what-is-iaas/) is like a company leasing a plot of land on which they can build whatever they want — but they need to provide their own building equipment and materials.

**Function-as-a-Service (FaaS)**: [FaaS](https://www.cloudflare.com/learning/serverless/glossary/function-as-a-service-faas/), also known as [serverless computing](https://www.cloudflare.com/learning/serverless/what-is-serverless/), breaks cloud applications down into even smaller components that only run when they are needed. Imagine if it were possible to rent a house one little bit at a time: for instance, the tenant only pays for the dining room at dinner time, the bedroom while they are sleeping, the living room while they are watching TV, and when they are not using those rooms, they don't have to pay rent on them

**Types of cloud:**

Cloud computing is a revolutionary technology transforming how we store, access, and process data. It simply refers to delivering computing resources, such as servers, storage, databases, software, and applications, over the Internet. Cloud computing uses a network of remote computer systems housed on the net to save and process data rather than relying on physical infrastructure.

**Public Cloud:**

* Public cloud services are provided by third-party providers over the internet. These services are available to multiple users and organizations on a pay-per-usage model.
* **Key Features**:
  + Scalability: Users can easily scale resources up or down based on demand.
  + Cost-effectiveness: Users only pay for the resources they consume.
  + Shared Infrastructure: Resources are shared among multiple users, leading to cost savings.
  + Accessibility: Services are accessible from anywhere with an internet connection.
* **Examples**: Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), IBM Cloud.

**Private Cloud:**

Private cloud is also known as an internal cloud or corporate cloud. It is used by organizations to build and manage their own data centers internally or by the third party. Private cloud infrastructure is exclusively used by a single organization

* **Key Features**:
  + Enhanced Security: Offers greater control over security measures and data privacy.
  + Customization: Allows organizations to tailor the infrastructure to meet specific needs and compliance requirements.
  + Performance: Provides predictable performance since resources are not shared with other organizations.
  + Control: Offers full control over infrastructure management and configuration.
* **Examples**: VMware Cloud Foundation, OpenStack, Microsoft Azure Stack.

**Hybrid Cloud:**

Hybrid Cloud is a combination of the public cloud and the private cloud.Hybrid cloud is partially secure because the services which are running on the public cloud can be accessed by anyone, while the services which are running on a private cloud can be accessed only by the organization's users. In a hybrid cloud setup, organizations can leverage the benefits of both public and private clouds to create a flexible and scalable computing environment.

* **Key Features**:
  + Flexibility: Organizations can move workloads between public and private clouds based on changing requirements.
  + Data Portability: Enables seamless data movement and interoperability between cloud environments.
  + Risk Mitigation: Provides redundancy and disaster recovery capabilities by distributing workloads across multiple environments.
  + Compliance: Helps organizations meet regulatory requirements by keeping sensitive data on-premises.
* **Examples**: AWS Outposts, Azure Arc, Google Anthos.

**Community Cloud:**

Community cloud allows systems and services to be accessible by a group of several organizations to share the information between the organization and a specific community. It is owned, managed, and operated by one or more organizations in the community, a third party, or a combination of them.

* **Key Features**:
  + Collaboration: Allows organizations within the same community to share resources and collaborate on projects.
  + Cost Sharing: Distributes the cost of infrastructure among multiple organizations, leading to cost savings.
  + Compliance: Offers specialized services and configurations to meet industry-specific regulatory requirements.
  + Security: Provides enhanced security measures tailored to the needs of the community.
* **Examples**: Government community clouds, healthcare community clouds, financial services community clouds.

**Multi Cloud:**

Multi-cloud is a strategy in cloud computing where companies utilize more than one cloud service provider or platform to meet their computing needs. It involves distributing workloads, applications, and statistics throughout numerous cloud environments consisting of public, private, and hybrid clouds.

* **Key Features**:
  + Vendor Diversification: Reduces dependency on a single cloud provider and mitigates the risk of vendor lock-in.
  + Best-of-Breed Solutions: Allows organizations to select the best-in-class services from different providers for each specific requirement.
  + Redundancy and Disaster Recovery: Provides redundancy by distributing workloads across multiple cloud environments.
  + Performance Optimization: Enables organizations to optimize performance by leveraging the strengths of different cloud providers.
* **Examples**: Using AWS for compute services, Azure for AI and machine learning, and GCP for data analytics.

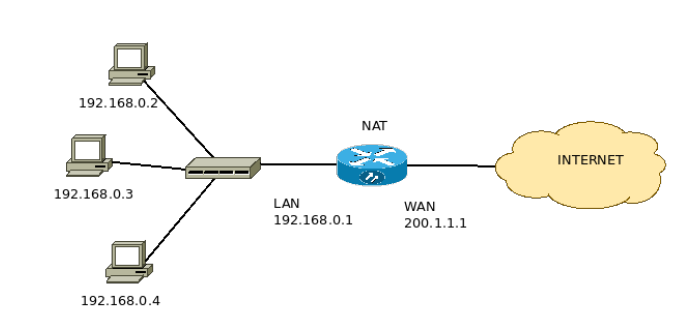
For details:

<https://www.javatpoint.com/types-of-cloud>

<https://www.geeksforgeeks.org/types-of-cloud/>

**NAT:**

NAT stands for Network Address Translation, is a technique used in networking to map one or more private IP addresses to a single public IP address. It primarily serves two purposes:



1. Conserving Public IP Addresses: With the limited availability of IPv4 addresses, NAT allows multiple devices within a private network to share a single public IP address. This conserves public IP address space, as only the NAT device needs a public IP address.
2. Enhancing Security: NAT acts as a barrier between the internal private network and the public internet. By hiding the internal IP addresses from external networks, it adds a layer of security, making it more difficult for attackers to directly target devices within the private network.

**For more info:**

[**https://www.geeksforgeeks.org/network-address-translation-nat/**](https://www.geeksforgeeks.org/network-address-translation-nat/)

**Software Management:-**

**1. Red Hat Package Manager(RPM)**

Red Hat Packet Manager (RPM) is a package management system used primarily in Red Hat Enterprise Linux (RHEL) and its derivatives, such as CentOS and Fedora. It is a command-line tool designed to simplify the process of installing, upgrading, and managing software packages on Linux systems.

Basic syntax: rpm [options] [package\_name]

Example : rpm -i package.rpm

Here,-i means install

1. Package Management: RPM manages software packages, which are collections of files and metadata necessary for installing and running software applications on a Linux system. These packages typically contain binaries, libraries, configuration files, and documentation.
2. Dependency Resolution: RPM handles dependencies, ensuring that required libraries and dependencies are installed along with the software package being installed. This helps in maintaining a consistent and functional software environment.
3. Installation and Removal: RPM facilitates the installation and removal of software packages using simple commands. Administrators can easily install or uninstall packages as needed, streamlining software management tasks.
4. Verification and Querying: RPM allows users to verify the integrity of installed packages and query the system to gather information about installed packages, including version numbers, dependencies, and file locations.
5. Package Building: RPM provides tools for building custom software packages from source code or existing binaries. This enables developers and system administrators to create custom packages tailored to their specific requirements.
6. Security: RPM includes features for digitally signing packages to ensure their authenticity and integrity. This helps prevent tampering and ensures that packages come from trusted sources.

**Advanced Package Tool(APT):**

APT, is a free-software user interface that works with core libraries to handle the installation and removal of software on Debian, and Debian-based Linux distributions.

APT simplifies the process of installing, updating, upgrading, and removing software packages on a system.

Here's a brief explanation of how APT works:

1. **Repositories**: APT relies on repositories, which are collections of software packages along with metadata about those packages. These repositories can be local or remote servers accessible over the internet.
2. **Package Index**: APT maintains an index of available packages within the repositories. This index includes information such as package names, versions, dependencies, and descriptions.
3. **Package Management Commands**: Users interact with APT using command-line tools such as ***apt-get*** and ***apt***. These commands allow users to perform various package management tasks, such as installing (***apt-get install***), removing (***apt-get remove***), updating (***apt-get update***), and upgrading (***apt-get upgrade***) packages.
4. **Dependency Resolution**: APT automatically resolves dependencies when installing or upgrading packages. It ensures that all required dependencies for a package are also installed, making it easier to manage software dependencies.
5. **Dependency Tracking**: APT keeps track of installed packages and their dependencies, ensuring that the system remains in a consistent state.
6. **Security Updates**: APT provides mechanisms for applying security updates to installed packages, helping to keep the system secure.

**tar, tgz and gz packages:**

Tar, tgz, and gz are file formats commonly used for archiving and compressing files and directories in Unix-like operating systems**.**

1. Tar (Tape Archive): A TAR file is what you'd call an archive, as it is only a collection of multiple files put together inside a single file. It doesn't compress the files by itself; it just bundles them together. Tar archives are commonly used for distribution or backup purposes.
2. TGZ (Tar + Gzip): TGZ is a compressed archive format that combines the functionality of Tar and Gzip. It first bundles files and directories into a single Tar archive and then compresses the Tar archive using the Gzip compression algorithm. TGZ files are often used to save disk space and facilitate faster file transfer.
3. GZ (Gzip): Gzip is a compression utility used to compress single files. It reduces the size of a file by replacing repetitive sequences of data with shorter representations. Gzip is frequently used in combination with Tar to compress Tar archives, resulting in TGZ files. However, Gzip can also be used independently to compress individual files

**cURL and wget:**

cURL:

Curl is a command-line tool and library used to transfer data to or from a server, supporting various protocols such as HTTP, HTTPS, FTP, FTPS, and more. It allows users to interact with web servers to download or upload files, make requests, and perform various network-related tasks. It can upload files as well as retrieve them.

Wget:

Wget is another command-line tool used for retrieving files from web servers using HTTP, HTTPS, and FTP protocols. Similar to Curl, Wget is commonly used to download files from the internet onto your local system. It supports recursive downloading, resuming interrupted downloads, and more. Its primary purpose is to download webpages---or even entire websites.

**User and Group management:-**

**useradd**: This command is used to create a new user on a Linux system. For example, to create a new user named "john", you would use:

*useradd john*

**groupadd**: This command is used to create a new group on a Linux system. For example, to create a new group named "developers", you would use:

*groupadd developers*

**usermod:**This command is used to modify an existing user account. For example, to add the user "john" to the "developers" group, you would use:

*usermod -aG developers john*

**groupmod**: This command is used to modify an existing group. For example, to change the name of the "developers" group to "engineers", you would use:

*groupmod -n engineers developers*

**userdel**: This command is used to delete a user account. For example, to delete the user "john", you would use:

*userdel john*

**groupdel**: This command is used to delete a group. For example, to delete the group "developers", you would use:

*groupdel developers*

**passwd**: This command is used to change a user's password. For example, to change the password for the user "john", you would use:

*passwd john*

**id**: This command is used to display user and group information. For example, to display information about the current user, you would use:

*id*

**whoami**: This command is used to display the current username. For example, to display the username of the current user, you would use:

*whoami*

***who****: This command is used to display information about currently logged in users. For example, to display a list of currently logged in users, you would use:*

*who*

**w**: This command is used to display information about currently logged in users, similar to the **who** command but with additional details. For example, to display detailed information about currently logged in users, you would use:

*w*

**last**: This command is used to display information about previous logins. For example, to display a list of previous login sessions, you would use:

*last*

**/etc/passwd :**

/etc/passwd file stores essential information, which is required during login i.e. user account information. /etc/passwd is a text file, that contains a list of the system’s accounts, giving for each account some useful information like user ID, group ID, home directory, shell, etc. It should have general read permission as many utilities, like ls use it to map user IDs to user names, but write access only for the superuser/root account.

Anshuman:x:1001:1001:John Doe:/home/Anshuman:/bin/bash

* Username: Anshuman
* UID: 1001
* GID: 1001
* Full Name: John Doe
* Home Directory: /home/Anshuman
* Default Shell: /bin/bash

**/etc/shadow file:**

The /etc/shadow file stores actual password in encrypted format for user’s account with additional properties related to user password i.e. it stores secure user account information. All fields are separated by a colon (:) symbol. It contains one entry per line for each user listed in /etc/passwd file.

Anshuman:$6$NAnoMEmP$GgRfy2.YYxwJNpFmb6c8sNZSb2ACwMvt4nd2HeVC6Mnb2cDyM3ly/F3Rgsq9sQuYTySR1qNmWb0IfmCZHs2.O/:18764:0:90:7:::

* Username: john
* Encrypted Password: $6$NAnoMEmP$GgRfy2.YYxwJNpFmb6c8sNZSb2ACwMvt4nd2HeVC6Mnb2cDyM3ly/F3Rgsq9sQuYTySR1qNmWb0IfmCZHs2.O/
* Last Password Change: 18764
* Minimum Password Age: 0
* Maximum Password Age: 90
* Password Warning Period: 7
* Account Expiration Date: (empty, indicating no expiration)

**/etc/group file:**

/etc/group is a text file which defines the groups to which users belong under Linux and UNIX operating system. Under Unix / Linux multiple users can be categorized into groups. Unix file system permissions are organized into three classes, user, group, and others. The use of groups allows additional abilities to be delegated in an organized fashion, such as access to disks, printers, and other peripherals. This method, amongst others, also enables the Superuser to delegate some administrative tasks to normal users.

[root@localhost ~]$ cat /etc/group

root:x:0:root

bin:x:1:root,bin,daemon

daemon:x:2:root,bin,daemon

sys:x:3:root,bin,adm

bar\_group:x:501:foo2,foo3

**Service Management:-**

Service management refers to the process of controlling, monitoring, and maintaining services, which are programs or daemons that run in the background and provide specific functionality on a computer system.

**Systemd :**

Systemd is like the traffic manager for your Linux system. It’s a modern tool that organizes and supervises how different parts of your system start and run. Imagine it as a smarter, faster way to handle tasks compared to older methods. Instead of doing things one after the other, systemd can get multiple tasks going at the same time, making your computer start up quicker and respond faster.

**systemctl and service commands :**

systemctl is a command-line utility used to manage systemd services. It allows users to control services by starting, stopping, restarting, enabling, disabling, and checking the status of services.

Syntax :

systemctl [command] [service]

To start the **nginx** service using systemctl:

*systemctl start nginx*

To check the status of the **nginx** service:

*systemctl status nginx*

To enable the **nginx** service to start automatically at boot time:

*systemctl enable nginx*

To restart the **nginx** service:

*systemctl restart nginx*

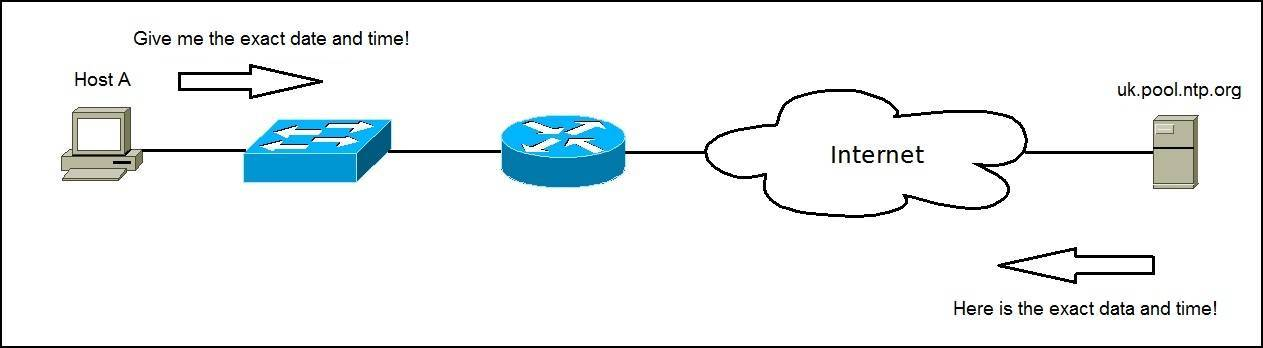
To stop the **nginx** service:

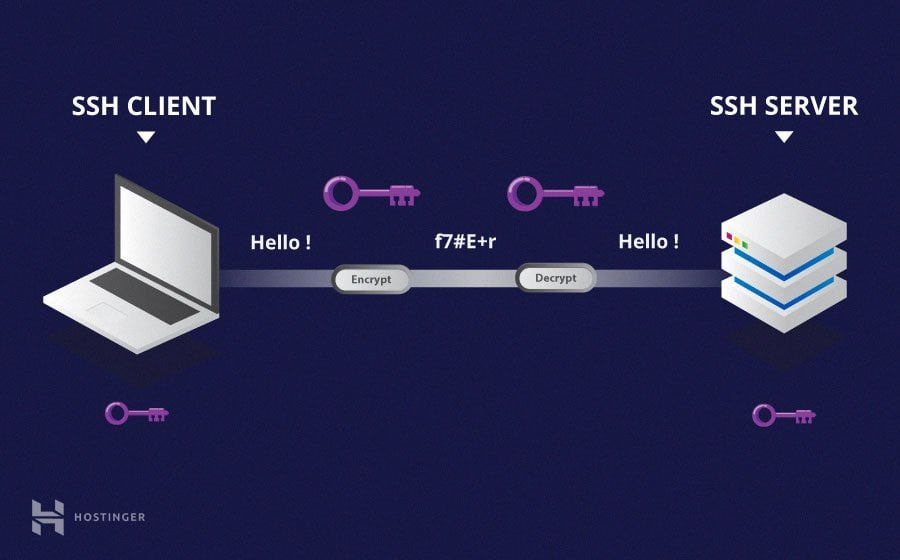
*systemctl stop nginx*

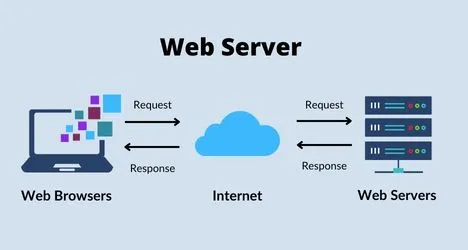
**Linux Servers :**

Linux servers are computer systems running on Linux operating systems that provide various services to clients or other systems on a network. These services can include network time synchronization, secure remote access, web hosting, certificate management, domain name resolution, IP address allocation, user authentication, traffic routing, data storage, email handling, load distribution, and more.

1. **Network Time Protocol (NTP)**: NTP servers synchronize the time of computer systems on a network. They ensure that all devices have accurate time settings, which is crucial for maintaining consistency in logs, security protocols, and various network operations. NTP uses a hierarchy of clocks and computers for synchronizing the current time.

For example, if there's a big differential between the time provider's time and the time on your local system, NTP will adjust the time on your local system in small increments until the time eventually becomes synchronized.

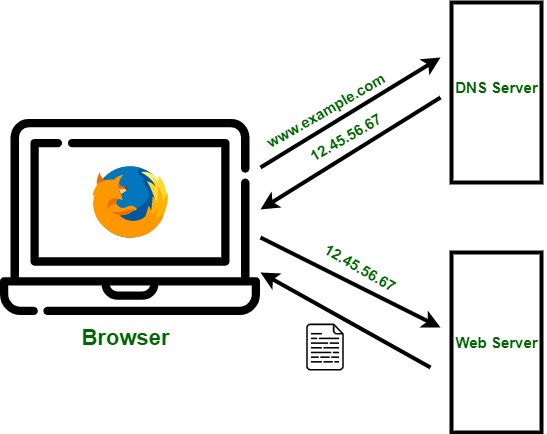
1. **Secure Shell (SSH)**: SSH servers provide secure remote access to a system's command-line interface or for secure file transfer. They use encryption to protect data transmitted over the network, allowing users to remotely manage and administer Linux servers securely. OpenSSH is an open source implementation of the Secure Shell (SSH) protocol and implemented by default on most Linux distributions.  
   Two major components of SSH include the SSH client and the SSH server. The SSH client is a program that is typically only run as needed. Once installed, the SSH server is a daemon that constantly runs in the background.
2. **Apache and NGINX servers**:

A web server is the program responsible for accepting HTTP requests from web browsers or clients and, in turn, sending the clients the files that form webpages. For example, webpages often consist of HTML documents and linked objects, such as images. A machine that has been dedicated to performing this role is also called a web server. Apache and NGINX are popular web servers that host and serve web content over HTTP or HTTPS protocols. They handle client requests, serve web pages, and can support various web technologies such as PHP, Python, and Ruby on Rails.

1. **Certificate Authority (CA)**: A digital certificate is an electronic document that can be used as proof of identification.CA servers issue digital certificates used for authenticating the identity of websites, servers, and users in SSL/TLS encrypted connections. They verify the authenticity of certificates and ensure secure communication over the internet.

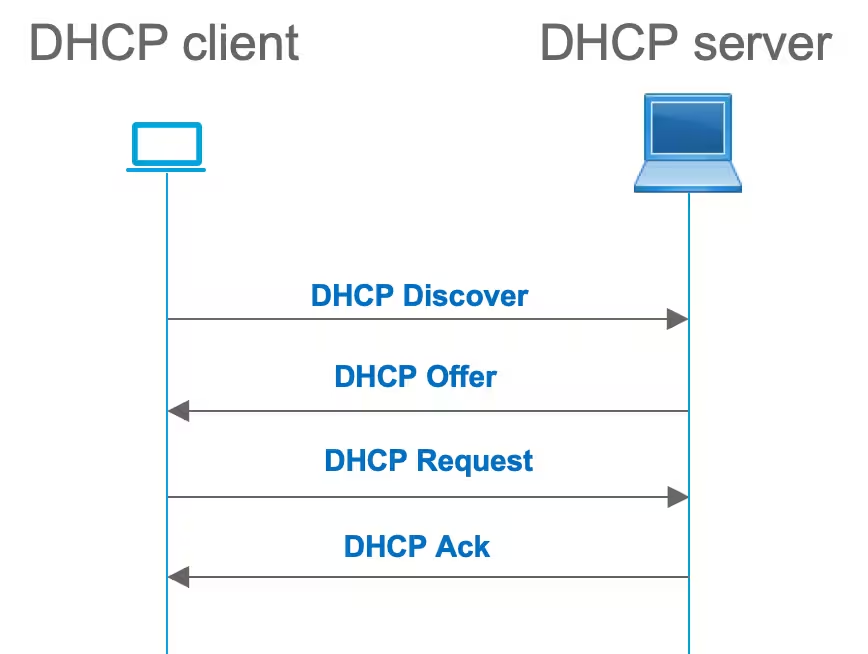
 For example, when using VPNs, you could use a digital certificate for authentication instead of a pre-shared key.

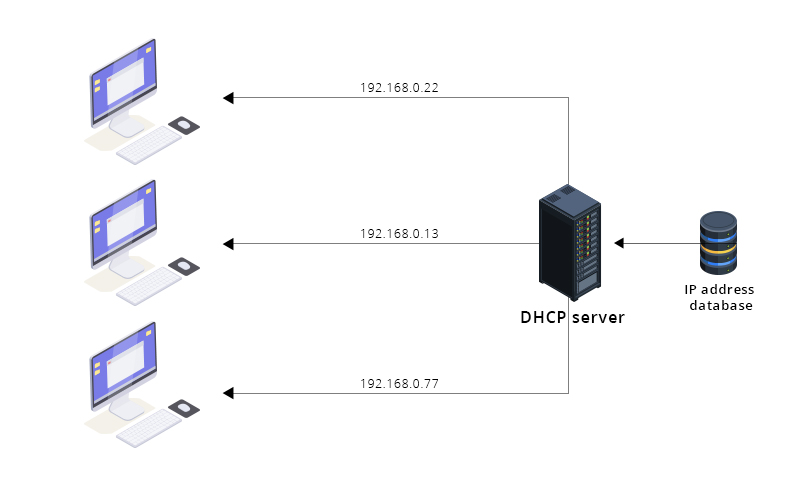
1. **Domain Name System (DNS)**: DNS servers convert domain names to IP addresses, allowing clients to access websites and services using human-readable names instead of numerical IP addresses. They maintain a distributed database of domain name mappings and facilitate efficient internet navigation.



1. **Dynamic Host Configuration Protocol (DHCP)**: DHCP servers automatically assign IP addresses, subnet masks, DNS servers, and other network configuration parameters to client devices on a network. They simplify network administration and ensure efficient IP address allocation.

DHCP also allows users who move from network to network to easily obtain an IP address appropriate for the subnet they are connected to. The DHCP server and the client use broadcasts to communicate with each other.

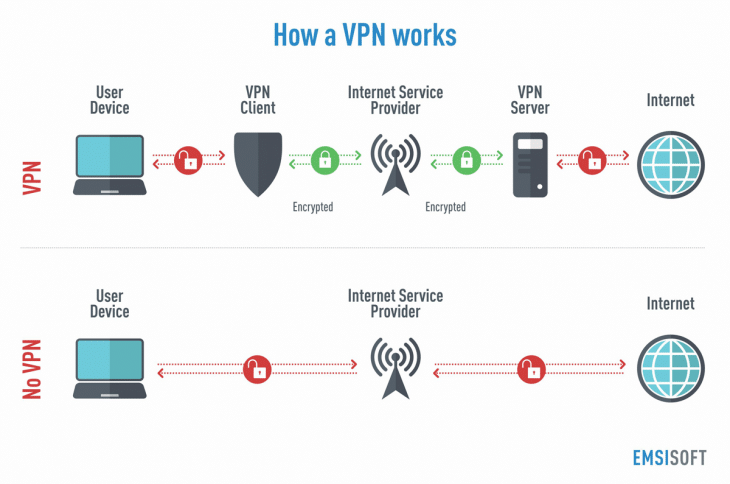




1. **Authentication Servers**: Authentication servers verify the identity of users and grant access to resources based on their credentials. They authenticate users against a central database such as LDAP (Lightweight Directory Access Protocol) or Active Directory. This is not only convenient for users, but also allows an administrator to monitor and audit user types and the type of access they have on each machine.
2. **Proxy Servers**:

A proxy is a computer that provides indirect internet access to the computers in your network. In most cases, a proxy server is installed on the same computer as the firewall. Proxy servers act as intermediaries between clients and other servers, intercepting and forwarding requests on behalf of clients. They can improve security, performance, and privacy by caching content, filtering requests, and hiding the client's IP address.

1. **Virtual Private Networks (VPN)**: VPN servers establish secure, encrypted connections between remote clients and a private network, allowing users to access resources securely over the internet. They provide remote access and secure communication for users working from remote locations.



1. **Monitoring Servers**: Monitoring refers to the process of monitoring the essential Linux services, including such things as operating system metrics, process state, logs, service state, and file system usage. It also refers to monitoring servers' availability. Monitoring servers collect and analyse data about the performance, availability, and health of network devices, servers, and applications. They generate alerts, reports, and visualizations to help administrators monitor and manage IT infrastructure effectively.
2. **Database Servers**: Database servers store and manage structured data, providing efficient storage, retrieval, and manipulation of information. They support various database management systems (DBMS) such as MySQL, PostgreSQL, MongoDB, and Oracle.
3. **Mail Servers**: Mail servers handle the sending, receiving, and storage of email messages. They use protocols like SMTP (Simple Mail Transfer Protocol), IMAP (Internet Message Access Protocol), and POP3 (Post Office Protocol version 3) to facilitate email communication.
4. **Load Balancers**: Load balancers distribute incoming network traffic across multiple servers or resources to ensure optimal resource utilization, high availability, and reliability. They enhance performance and scalability by evenly distributing the workload among server instances.

**Scheduling and Automation:**

Scheduling and automation are essential aspects of system administration that involve automating repetitive tasks, executing commands at specific times, and managing processes efficiently. Here's an explanation of various tools and commands used for scheduling and automation:

**cron:**

Cron is a time-based job scheduler in Unix-like operating systems. It allows users to schedule tasks (known as cron jobs) to run periodically at specified intervals, such as minutes, hours, days, weeks, or months. Cron jobs are defined in crontab (cron table) files, which list the commands or scripts to be executed along with their schedule.

Example:

Let's say you want to schedule a script named backup.sh to run every day at 2:00 AM. You can do this by adding an entry to your crontab file using the crontab -e command:

*0 2 \* \* \* /path/to/backup.sh*

**Job control commands**: Job control commands allow users to manage processes running in the background or in the foreground within a shell session. These commands are typically used in interactive shell sessions or in shell scripts to control the execution of multiple commands or processes.

* **bg**: Move a stopped or background job to the background.
* **fg**: Bring a background job to the foreground.
* **jobs**: List the current jobs running in the background.
* **ctrl+z**: Suspend a foreground job.
* **ctrl+c**: Terminate a foreground job

**kill command:**

The kill command is used to terminate processes by sending them signals. It allows users to gracefully shut down or forcefully terminate processes based on their requirements.

Example:

Suppose you have a process with PID (Process ID) 1234 that you want to terminate. You can use the kill command with the PID as follows:

*$ kill 1234*