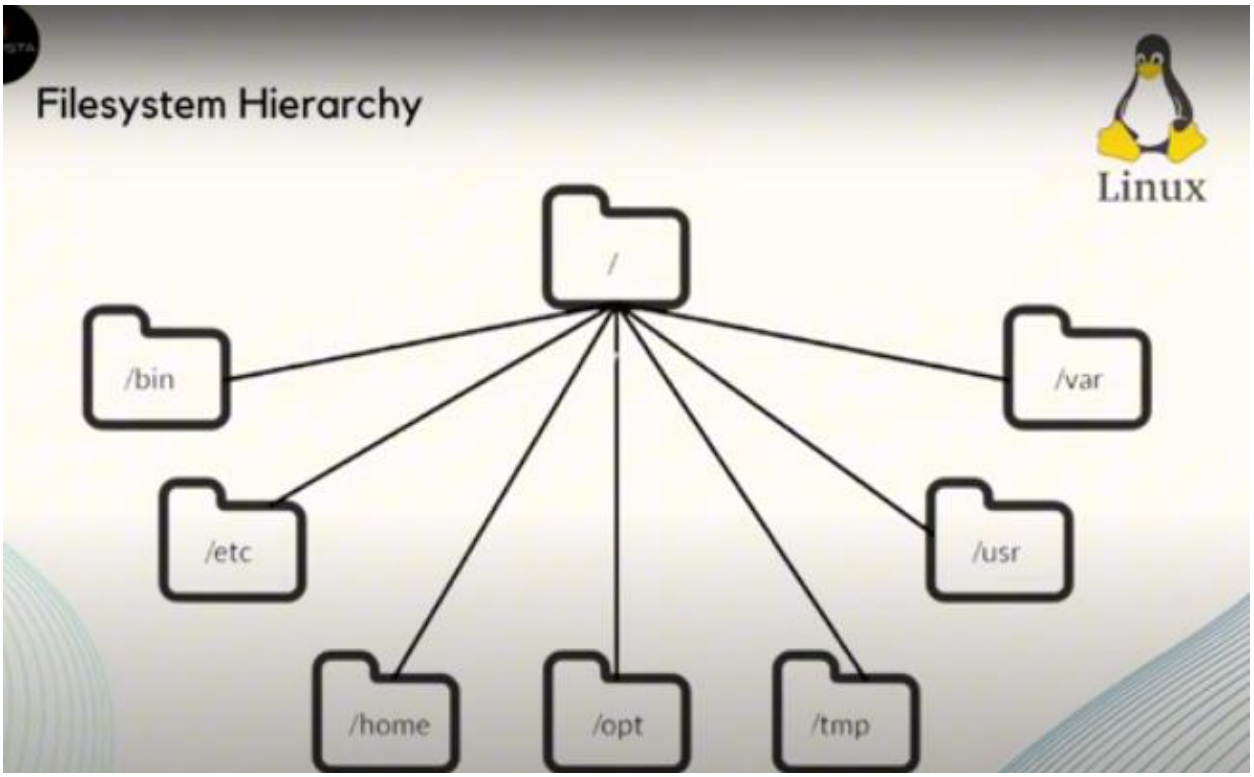
Filesystem Hierarchy

The file system in Ubuntu, like in most Unix-like operating systems, is structured hierarchically
Here's an overview of its key components:

- **Root Directory (/)**
  - The root directory is the top-level directory of the file system hierarchy
  - All other directories and files are located beneath it
  - It is represented by a forward slash (/)



Filesystem Hierarchy

- **/bin**
  - Essential command binaries (e.g., ls, cp, mv, rm)
- **/boot**
  - Static files of the boot loader (e.g., kernels, initrd).
- **/dev**
  - Device files (e.g., /dev/sda, /dev/null).
- **/etc**
  - Configuration files.
- **/home**
  - Home directories for users.
- **/lib**
  - Essential shared libraries and kernel modules
- **/media**
  - Mount points for removable media (e.g., USB drives).

Linux

Filesystem Hierarchy

- **/mnt**
  - Temporary mount points.
- **/opt**
  - Optional application software packages.
- **/proc**
  - Virtual filesystem providing process and system information.
- **/sbin**
  - Essential system binaries.
- **/tmp**
  - Temporary files.
- **/usr**
  - Secondary hierarchy for read-only user data

What we will learn today

File Operations Commands

man
mkdir
touch
mv
cp
rm

Man (Manual)

The man command in Linux is used to format and display the manual pages for commands

- It stands for "manual,"
- It provides detailed information about various commands

Man <Command>

Example

- man ls
- man mkdir
- man grep: View the manual page for the grep command.
- man -f passwd: Find brief descriptions of all sections related to passwd.

File operations

In Linux, file operations are managed using various commands. Here are some commonly used ones:

- **mkdir** (Make Directory)
  - **Syntax:** mkdir [options] directory(s)
  - **Example:** mkdir new_directory creates a new directory named "new_directory"
- **Description**
  - Create the DIRECTORY(ies), if they do not already exist.
  - Mandatory arguments to long options are mandatory for short
- **Options**
  - **-m**, --mode=MODE
  - set file mode (as in chmod), not a=rwx - umask
  - **-p,** --parents
  - no error if existing, make parent directories as needed,
  - with their file modes unaffected by any -m option.
  - **-v,** --verbose
  - print a message for each created directory

File operations

- **touch** (Create/Update files)
  - **Syntax:** touch [options] file(s)
  - **Example:** touch newfile.txt creates a new empty file "newfile.txt" or update it
- **Description**
  - Update the access and modification times of each FILE to the current time.
  - A File argument that does not exist is created empty, unless -c or -h is supplied.
  - **-a** change only the access time
  - **-f** (ignored)
  - **-m** change only the modification time

File operations

- **mv** (Move/Rename)
  - **Description:** Moves (or renames) files or directories from source to destination
  - **Syntax:** mv [options] source destination
  - **Example:** mv file.txt newname.txt renames "file.txt" to "newname.txt" in the same directory
- **Description**
  Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.
- **-b**
  - like --backup but does not accept an argument
- **--debug**
  - explain how a file is copied. Implies -v
- **--exchange**
  - exchange source and destination
- **-f, --force**
  - do not prompt before overwriting
- **-i**, --interactive
  - prompt before overwrite
- **-n**, --no-clobber
  - do not overwrite an existing file

File operations

- **cp** (Copy)
  - **Syntax:** cp [options] source destination
  - **Example:** cp file.txt /path/to/destination/ copies "file.txt" to the specified destination directory
- **Description** Copies files or directories from a source location to a destination location. It preserves the original file's permissions,
- Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.
- **-b** like --backup but does not accept an argument
- **-f,** --force
  - if an existing destination file cannot be opened, remove
  - it and try again (this option is ignored when the -n option is also used)
- **-i,** --interactive
  - prompt before overwrite (overrides a previous -n option)
- **-v,** --verbose
  - explain what is being done

File operations

- **rm** (Remove/Delete)
  - **Description:** Removes (deletes) files or directories. Be cautious as rm permanently deletes files and directories, and there is no easy way to recover them.
  - **Syntax:** rm [options] file(s)/directory(s)
  - **Example:** rm file.txt removes "file.txt" from the file system
- **Description**
- This manual page documents the GNU version of rm. rm removes each specified file.
- **-f,** --force
  - ignore nonexistent files and arguments, never prompt
- **-i** prompt before every removal
- **-I** prompt once before removing more than three files, or when
  - removing recursively: less intrusive than -i, while still giving protection against most mistakes

What we will learn today

Viewing the file content

cat
less
more
tail
head

Linux

Viewing the file content

In Linux, several commands are used to view the contents of files. Here are some commonly used ones:

Linux

- **cat** (Concatenate and display file content)
  - **Syntax:** cat [options] file
  - **Example:** cat file.txt
  - **Description:** Displays the entire content of the file at once

The cat command has several options that modify its output. Commonly used options are

- **Description:**
  - Concatenate FILE(s) to standard output. With no FILE, or when FILE is -, read standard input.
  - **-A,** --show-ala equivalent to -vET
  - **-b,** --n number nonempty output lines, overrides -n
  - **-u** (ignored) -v, --show-nonprint use and M- notation, except for LFD and TAB

Viewing file content

- **less** (View file content interactively with scrolling)
  - **Syntax:** less file
  - **Example:** less file.txt

- **Description:** Allows you to scroll through the file content using arrow keys. Press q to quit
  - **-?**
    - Display help information.
  - **-e**
    - Exit less after reaching the end of the file.
  - **-E**
    - Exit less after reaching the end of the file twice.
  - **-f**
    - Force opening of non-regular files.
  - **-g**
    - Highlight only the last search match.

Viewing file content

- **more** (View file content page by page)
  - **Syntax:** more file
  - **Example:** more file.txt

- **Description:** Similar to less, but with fewer features. Press space to go to the next page and q to quit
  - **-l,** --logical
    - Do not pause after any line containing a ^L (form feed).
  - **-f,** --no-pause
    - Count logical lines, rather than screen lines (i.e., long lines are not folded).
  - **s,** --squeeze
    - Squeeze multiple blank lines into one.
  - **-u,** --plain
    - Suppress underlining. This option is silently ignored as
    - backwards compatibility.
  - **-n,** --lines number
    - Specify the number of lines per screenful.

Viewing file content

- **tail** (Display the end of a file)
  - **Syntax:** tail [options] file
  - **Example:** tail file.txt
  - **Options:** -n [number]: Specify the number of lines to display
  - **Example:** tail -f file.txt

- **Description:** By default, shows the last 10 lines of the file
  - **-n,** --lines=[+]NUM
    - output the last NUM lines, instead of the last 10; or use
    - -n +NUM to skip NUM-1 lines at the start
  - **--pid**=PID
    - with -f, terminate after process ID, PID dies; can be repeated to watch multiple processes
  - **--retry**
    - keep trying to open a file if it is inaccessible
  - **-v,** --verbose
    - always output headers giving file names

Viewing file content

- **head** (Display the beginning of a file)
  - **Syntax:** head [options] file
  - **Example:** head file.txt
  - **Description:** By default, shows the first 10 lines of the file
  - **Options:** -n [number]: Specify the number of lines to display
  - **Example:** head -n 5 file.txt

- **options**

- **-n,** --lines=[-]NUM
  - print the first NUM lines instead of the first 10; with the leading '-', print all but the last NUM lines of each file

- **-q,** --quiet, --silent
  - never print headers giving file names

- **-v,** --verbose
  - always print headers giving file names

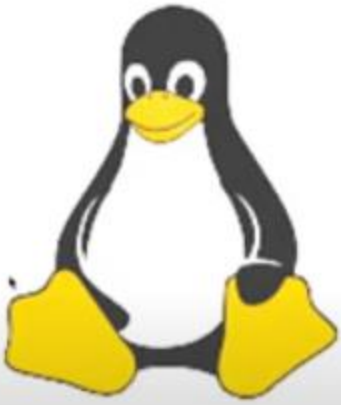
- **-z,** --zero-terminated
  - line delimiter is NUL, not newline

What we will learn today

Navigating the file system

- cd
- ls
- pwd

Pipes in linux



Linux

Navigating the file system

Linux

- **cd** (Change Directory)
  - Used to change your current directory
  - **Syntax:** cd [directory]
  - **Example:** cd Documents changes the current directory to the "Documents" folder

Description

- **cd :**
  - Change to the home directory.
- **cd .. :**
  - Move up one directory (to the parent directory).
- **cd - :**
  - Switch to the previous directory.
- **cd / :**
  - Change to the root directory.
- **cd ~ :**
  - Change to the home directory.
- **cd ~**
  - username : Change to the home directory of the specified user.
- **cd /path/to/directory :**
  - Change to a specified directory by providing its path.

Navigating the file system

Navigating the file system in Ubuntu involves using three fundamental commands: cd, ls, and pwd

- **ls** (List Directory Contents)
  - Lists the files and directories in the current directory
  - **Syntax:** ls [options] [directory]
  - **Example:** ls -l lists the contents of the current directory in long format

Description:

- **-a,** --all
  - do not ignore entries starting with .
- **--block-size**=SIZE
  - with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below
- **-B,** --ignore-backups
  - do not list implied entries ending with ~
- **-d,** --directory
  - list directories themselves, not their contents
- **-f** do not sort, enable -aU, disable -ls --color

Navigating the file system

Navigating the file system in Ubuntu involves using three fundamental commands: cd, ls, and pwd

- **pwd** (Print Working Directory)
  - Prints the full path of the current working directory
  - **Syntax:** pwd
  - **Example:** Typing pwd will display something like /home/username/Documents

Description

- **-L,** --logical
  - use PWD from environment, even if it contains symlinks
- **-P,** --physical
  - avoid all symlinks
- **--help** display this help and exit
- **--version**
  - output version information and exit

Pipes in linux

In Linux, a pipe (|) is used to pass the output of one command as the input to another, allowing for complex command sequences

- command1 | command2
  - **List and Filter Files:**
    - ls | grep report
  - **Search File Content:**
    - cat logfile.txt | grep error
  - **Find Processes:**
    - ps aux | grep ssh
  - **Count Matching Lines:**
    - grep error logfile.txt | wc -l
  - **Disk Usage for Root Filesystem:**
    - df -h | grep '/$' | awk '{print $4}'

What we will learn today

File Permissions

- Understanding file Permission
- Type of Files
- Changing file Permission

File Permissions

Understanding File Permissions

- **Overview:** Permissions determine who can read, write, or execute a file
- **Types of Permissions:**
  - **Read (r):** Allows viewing the contents of the file
  - **Write (w):** Allows modifying the file
  - **Execute (x):** Allows running the file as a program
- **Permission for:**
  - **Owner:** The user who owns the file
  - **Group:** Other users in the file's group
  - **Others:** All other users

File Permissions

Types of Files:

- **Regular File:**
  - Normal files that contain data, such as text file or binary file
  - Example: **-rw-r--r--**
  - **-** indicates a regular file
  - **rw-** means the owner can read and write
  - **r--** means the group can read
  - **r--** means others can read
- **Directory:**
  - A folder in which you store files
  - Example: **drwxr-xr-x**
  - **d** indicates a directory
  - **rwx** means the owner can read, write, and access the directory
  - **r-x** means the group can read and access the directory
  - **r-x** means others can read and access the directory
- **Links:**
  - Pointers to other files or directories
  - Example: **lrwxrwxrwx**
  - **l** indicates a symbolic link

ions

File Permissions

Changing File Permissions (chmod)

Step-by-Step Guide:

- **Check Current Permissions:**
  - Use **ls -l filename** to see the current permissions

- **Change Permissions Using Symbolic Mode:**
  - **chmod u+x filename** adds execute permission for the owner
  - **chmod g-w filename** removes write permission for the group
  - **chmod o+r filename** adds read permission for others

Change Permissions Using Numeric Mode

- **chmod 755 filename** sets permissions so
  - the **owner** can read, write, and execute
  - while the **group** and **others** can read and execute
- **The number represents permissions in binary**
  - **4** for read
  - **2** for write
  - **1** for execute

What we will learn today

Understanding File Ownership

- File ownership
- File Group
- Changing file ownership
- Changing file group
- Changing both

Linux

Understanding File Ownership and Group in Linux

File Ownership

- **Owner**
  - Each file and directory in Linux has an owner, typically the user who created it
- **Importance**
  - The owner has special permissions, usually including the ability to read, write, and execute the file
- **Usage**
  - Changing the ownership of a file can transfer these permissions to another user, useful for managing access in collaborative environments

File Group

- **Group**
  - Files and directories are also associated with a group. A group is a collection of users
- **Importance**
  - Group permissions allow users within the same group to share access to files, enabling collaborative work without changing individual ownership
- **Usage**
  - Changing the group ownership allows different sets of users to gain or lose access to the file based on their group membership

Change Ownership

Changing File Ownership (chown, chgrp)

Change Owner

- Command: **chown newowner filename**
- **Example:**
  - **chown zafar myfile.txt**
- This command changes the owner of myfile.txt to zafar.
  - **Before:** -rw-r--r-- 1 **babarzahoor clouddev** 1234 Jun 12 10:00 **myfile.txt**
  - **After:** -rw-r--r-- 1 **zafar clouddev** 1234 Jun 12 10:00 **myfile.txt**

Change Group

- Command: **chgrp newgroup filename**
- **Example:**
  - **chgrp owb myfile.txt**
- This command changes the group of myfile.txt to developers.
  - **Before:** -rw-r--r-- 1 **zafar clouddev** 1234 Jun 12 10:00 **myfile.txt**
  - **After:** -rw-r--r-- 1 **zafar owb** 1234 Jun 12 10:00 **myfile.txt**

- **Change Both Owner and Group**
  - **Command:** chown newowner:newgroup filename
  - **Example:**
    - chown kiramat:clouddev myfile.txt
  - This command changes both the owner to alice and the group to developers for myfile.txt
  - **Before:** -rw-r--r-- 1 zafar owb 1234 Jun 12 10:00 myfile.txt
  - **After:** -rw-r--r-- 1 kiramat clouddev 1234 Jun 12 10:00 myfile.txt

Managing Users and Groups in Linux

Overview

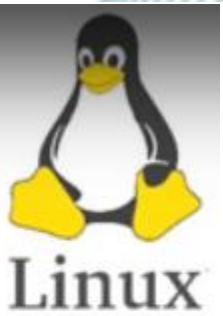
- Linux uses a multi-user system where multiple users can operate simultaneously
- **Users:** Individual accounts with specific permissions and settings
- **Groups:** Collections of users that can share permissions and access levels

Importance

- **Security:** Proper management ensures that only authorized users have access to sensitive data and system functions
- **Organization:** Helps in maintaining a structured system, making it easier to manage permissions and resources
- **Resource Allocation:** Facilitates efficient allocation of system resources by grouping users with similar requirements
- **Accountability:** Tracks user actions and ensures accountability by assigning unique user accounts

Linux!

Managing Users and Groups in Linux



Adding, Setting Password and Modifying Users

Adding Users

- Command: adduser
  - Example:
    - **sudo adduser babarzahoor**
    - **sudo adduser aliabdullah**

Set Password

- After creating the user with adduser, you can set the initial password for the user using the
- command: passwd
  - Example:
    - **sudo passwd cloud123**
    - **sudo passwd owb123**

n Linux!

Managing Users and Groups in Linux

Modifying Users:

- To change the username of an existing user:
  - Command: **usermod**
- Example
  - **sudo usermod -l newname oldname**
  - **sudo usermod -l ali babarzahoor**

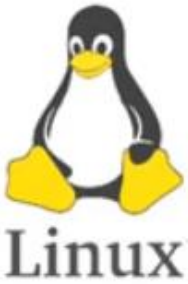
- **Change home directory:** -d /new/home
  - Example:
    - **sudo usermod -d /new/home babarzahoor**

Adding User to Additional Groups

How to Add User to Additional Groups?

- To add a user to additional groups:
  - **sudo usermod -aG groupname username**
  - Example
    - **sudo usermod -aG clouddev babarzahoor**

Managing Users and Groups in Linux



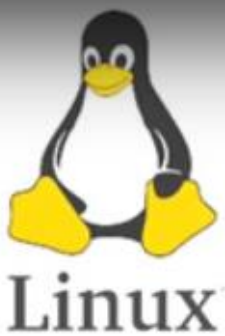
Basic Deletion

- To delete a user account without removing the home directory or any files
  - **sudo userdel username**
- Replace username with the username of the user you want to delete
  - Example:
    - **sudo userdel babarzahoor**
- This command deletes the user account babarzahoor, but leaves the home directory and user files intact

Delete User and Home Directory:

- To delete a user account and remove the home directory
  - **sudo userdel -r username**
  - Example:
    - **sudo userdel -r babarzahoor**
- This command deletes the user account babarzahoor and removes the home directory /home/babarzahoor

Managing Users and Groups in Linux



Modifying Users:

- To change the username of an existing user:
  - Command: **usermod**
- Example
  - **sudo usermod -l newname oldname**
  - **sudo usermod -l ali babarzahoor**

- **Change home directory:** -d /new/home
  - Example:
    - **sudo usermod -d /new/home babarzahoor**

Adding User to Additional Groups

How to Add User to Additional Groups?

- To add a user to additional groups:
  - **sudo usermod -aG groupname username**
  - Example
    - **sudo usermod -aG clouddev babarzahoor**

What we will study today

Managing Users in Linux (PART 2)

- Lock User
- Unlock User

Managing Groups in Linux

- Creating Group
- Creating Group with specific GID
- Change Group name
- Deleting Group

Cronjobs of the user

Mail spool

agement!

Managing Users and Groups in Linux

Lock and Unlock User Account

Lock User Account:

- **sudo usermod -L username**
- Replace username with the username of the user
  - Example:
    - **sudo usermod -L babarzahoor**
- This command locks the account of user babarzahoor, preventing them from logging in

Unlock User Account:

- **sudo usermod -U username**
- Replace username with the username of the user
- Example:
  - **sudo usermod -U babarzahoor**
- This command unlocks the account of user babarzahoor, allowing them to log in again

agement!

Managing Groups in Linux

Creating Groups

- Create a new group
  - Command:
    - **sudo groupadd groupname**
  - Example:
    - **sudo groupadd clouddev**

Create a new group with a specific GID

- Command:
  - **sudo groupadd -g GID groupname**
- Example:
  - **sudo groupadd -g 1001 clouddev**

agement!

Managing Users Cronjobs and mail spool

- **Cron Jobs:**
  - If the user has scheduled cron jobs, you should manually check and remove any associated cron jobs
  - Cron jobs are typically found in **/var/spool/cron/crontabs/username**
  - Example:
    - **sudo rm /var/spool/cron/crontabs/babarzahoor**
  - This command removes any cron jobs associated with babarzahoor

- **Mail Spool:**
  - Ensure the mail spool is removed if not handled by the userdel command
  - Mail spool files are located in **/var/mail/username**
  - Example:
    - **sudo rm /var/mail/babarzahoor**
  - This command removes the mail spool file for babarzahoor

Network

A network is a collection of devices (computers, servers, printers, etc.) interconnected to share resources and communicate with each other. Networks can be categorized based on their size and geographic scope:

- **LAN (Local Area Network):** Limited to a small geographic area like a home, office, or building.
- **WAN (Wide Area Network):** Spans large distances, often connecting LANs across cities or countries.

IP address

An IP (Internet Protocol) address is a unique numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. IP addresses serve two main purposes in networking:

1. **Identifying Devices**
   - An IP address uniquely identifies a device on a network, allowing other devices to locate and communicate with it. Think of it like a phone number or street address for devices on the internet or within a local network.

2. **Routing Data**
   - IP addresses are used by routers to determine the best path for data packets to travel from the source device to the destination device across interconnected networks. Routers use IP addresses to forward packets efficiently and ensure data reaches its intended recipient.

Types of IP Addresses

There are two primary versions of IP addresses currently in use:

1. IPv4 (Internet Protocol version 4)
- Consists of a 32-bit address, typically expressed as four decimal numbers separated by dots (e.g., 192.168.1.1).
- IPv4 addresses are currently the most widely used type of IP address but are limited in number, which has led to the adoption of IPv6.

2. IPv6 (Internet Protocol version 6)
- Uses a 128-bit address, expressed as eight groups of hexadecimal numbers separated by colons (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334).
- IPv6 addresses were developed to address the exhaustion of IPv4 addresses and provide a much larger pool of unique addresses.

Subnetting

Subnetting is a technique used in networking to divide a single large network into smaller, more manageable sub-networks or subnets. This process helps optimize network performance, improve security, and efficiently allocate IP addresses. Here's an explanation of subnetting along with tables to illustrate the concept:

Subnetting Explanation

- **Purpose of Subnetting:**
  - Efficient Use of IP Addresses: Instead of assigning a large block of IP addresses to a single network, subnetting allows for the creation of smaller, more focused sub-networks.
- **Improved Performance:**
  - Smaller subnets can reduce network congestion and improve data transfer speeds by isolating traffic within specific segments of the network.
- **Enhanced Security:**
  - Subnets can be used to logically group devices based on their roles or departments, allowing for better control and monitoring of network traffic.

IPv4 Address Classes

Class A

- **Range:** 0.0.0.0 to 127.255.255.255
- **First Octet:** The first octet (8 bits) of a Class A address starts with 0, allowing for 128 possible networks.
- **Network Portion:** First octet identifies the network, and the remaining three octets identify hosts.
- **Hosts per Network:** Supports up to 16,777,214 hosts per network.
- **Example:** 10.0.0.0/8 (10.0.0.0 to 10.255.255.255)

Class B

- **Range:** 128.0.0.0 to 191.255.255.255
- **First Octet:** The first octet of a Class B address starts with 10, allowing for 16,384 possible networks.
- **Network Portion:** First two octets identify the network, and the remaining two octets identify hosts.
- **Hosts per Network:** Supports up to 65,534 hosts per network.
- **Example:** 172.16.0.0/16 (172.16.0.0 to 172.31.255.255)

IPv4 Address Classes

Class C

- **Range:** 192.0.0.0 to 223.255.255.255
- **First Octet:** The first octet of a Class C address starts with 110, allowing for 2,097,152 possible networks.
- **Network Portion:** First three octets identify the network, and the last octet identifies hosts.
- **Hosts per Network:** Supports up to 254 hosts per network.
- **Example:** 192.168.0.0/24 (192.168.0.0 to 192.168.255.255)

Class D (Multicast)

- **Range:** 224.0.0.0 to 239.255.255.255
- **Purpose:** Reserved for multicast addressing, used for one-to-many communication where data is sent from one sender to multiple receivers simultaneously.

Class E (Experimental)

- **Range:** 240.0.0.0 to 255.255.255.255
- **Purpose:** Reserved for experimental and research purposes, not used for general addressing or networking.

What we will learn today

Networking Basics

Configuration Files

Linux

Networking Basics

- Linux is a powerful operating system widely used for
  - Servers
  - Network devices
  - Some desktops due to its robust networking capabilities

- **Network Interfaces:**
  - A network interface is the hardware component (like ethernet card or wifi adapter) that connects your device to a network
  - Linux assigns a name to each interface, typically starting with "eth" for ethernet (e.g., eth0) or "wlan" for wifi (e.g., wlan0)

Configuration Files:

Network configuration files define how your Linux device interacts with the network

Common configuration files include:

- **/etc/hosts**
  - Local hostname to IP address mapping for faster lookups
- **/etc/resolv.conf**
  - Stores IP addresses of DNS servers for translating domain names to IPs
- **/etc/sysconfig/network-scripts/ (Linux specific)**
  - Holds individual interface configuration files (e.g., ifcfg-eth0)
  - These define settings like IP address, subnet mask, and gateway

/etc/hosts

Purpose

- Maps hostnames to IP addresses for faster local lookups, bypassing DNS queries

Format

- **Each line contains an IP address followed by one or more hostnames**

Example

- vim /etc/host

```
127.0.0.1   localhost
192.168.1.100   myserver.localdomain   myserver
```

Usage

- Useful for local development, small networks, and quick testing
- Prioritized over DNS for hostname resolution

/etc/resolv.conf

Purpose:

- This file contains the IP addresses of DNS servers that the system uses to translate domain names into IP addresses

Common Entries:

- nameserver: Specifies the IP address of a DNS server

search:

- Specifies the search domains to append to hostnames

options:

- Allows you to set various resolver options

Example: Vim /etc/resolv.conf

```
nameserver 8.8.8.8
nameserver 8.8.4.4
search example.com
```

etc/sysconfig/network-scripts/ (Linux specific)

Purpose:

- This directory contains configuration files for each network interface on the system. These files are typically named ifcfg-<interface> (e.g., ifcfg-eth0) and define settings such as the IP address, subnet mask, gateway, and more

Common Parameters:

- DEVICE: The name of the device (e.g., eth0)

BOOTPROTO:

- The method used to assign the IP address (static or dhcp)

ONBOOT:

- Specifies whether the interface should be activated at boot (yes or no)

IPADDR:

- The static IP address assigned to the interface

NETMASK:

- The subnet mask

GATEWAY:

- The default gateway


```
DEVICE=eth0
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.1.100
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
```

Netplan

Netplan is a utility for easily configuring networking on a Linux system. It uses YAML descriptions of the network configuration to generate configurations for various network renderers.

Key Netplan Commands

- Apply Configuration:
  - **sudo netplan apply**
- Generate Configuration Files:
  - **sudo netplan generate**
- Test Configuration:
  - **sudo netplan try**
- View Netplan Status:
  - **sudo netplan status**

Basic YAML Configuration

Below is a simple YAML configuration file for Netplan, typically located in **/etc/netplan/** directory:

```yaml
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: true
```

Example for Static IP Configuration

What we will learn today

Basic networking commands

- ping
- ip
- ifconfig
- netstat
- ss

Linux

Basic networking commands

Linux provides a variety of networking commands to manage and troubleshoot network connections. Here are some basic and commonly used commands:

ping	Test connectivity to a remote host
ifconfig	Configure network interfaces (deprecated)
ip	Manage network interfaces, IP addresses, and routes
netstat	Display network connections and statistics (deprecated)
ss	Display socket statistics and network connections

Basic networking commands

ping

The ping command is used to test connectivity between two nodes on a network by sending ICMP echo request packets and waiting for a reply

- Syntax: **ping [options] destination**
- **Examples:**
- Ping a hostname
  - **ping www.google.com**
- Ping an IP address
  - **ping 8.8.8.8**
- Send a specific number of packets
  - ping -c 4 www.google.com
- Specify packet size
  - **ping -s 56 www.google.com**

Basic networking commands

ifconfig

The ifconfig command is used to configure network interfaces, however it is considered deprecated and replaced by the ip command in modern Linux distributions

- Syntax: **ifconfig [interface] [options]**

Examples:

- Display all network interfaces
  - **ifconfig**
- Display information about a specific interface
  - **ifconfig eth0**
- Assign an IP address to an interface
  - **sudo ifconfig eth0 192.168.1.100**
- Bring an interface up
  - **sudo ifconfig eth0 up**
- Bring an interface down
  - **sudo ifconfig eth0 down**

Basic networking commands

ip [object] [command] [options]

Examples:

- Display all network interfaces and their IP addresses
  - **ip addr show**
- Display detailed information about a specific interface
  - **ip addr show eth0**
- Assign an IP address to an interface
  - **sudo ip addr add 192.168.1.100/24 dev eth0**
- Remove an IP address from an interface
  - **sudo ip addr del 192.168.1.100/24 dev eth0**
- Bring an interface up
  - **sudo ip link set eth0 up**
- Bring an interface down
  - **sudo ip link set eth0 down**
- Display the routing table
  - **ip route show**
- Add a static route
  - **sudo ip route add 192.168.2.0/24 via 192.168.1.1**
- Delete a static route
  - **sudo ip route del 192.168.2.0/24**

Basic networking commands

netstat

The netstat command is used to display network connections e.g:

- routing tables
- interface statistics
- masquerade connections
- multicast memberships

It is considered deprecated and replaced by the ss command in modern Linux distributions

- Syntax: **netstat [options]**

Examples:

- Display all active connections
  - netstat
- Display all listening ports
  - netstat -l
- Display TCP connections
  - netstat -t
- Display UDP connections
  - netstat -u
- Display routing table
  - netstat -r
- Display network interface statistics
  - netstat -i

Basic networking commands

ss

The ss command is used to display socket statistics & it's a modern replacement for netstat

- Syntax: **ss [options]**

Examples:

- Display all open network connections
  - ss -a
- Display all listening ports
  - ss -l
- Display all TCP connections
  - ss -t
- Display all UDP connections
  - ss -u
- Display network statistics
  - ss -s
- Display processes using network connections
  - ss -p

What we will learn today

Monitoring

- htop
- atop
- iftop
- nload

Linux

mands!

htop:

- **Usage:**
  - htop is an interactive process viewer for Unix systems, it provides a more user-friendly and colorful interface compared to the traditional top command
- **Features**
  - Allows you to view and manage running processes, monitor CPU, memory, and swap usage, and interactively kill processes
- **Installation**
  - Usually available in Linux distributions through package managers (e.g., apt, yum)
  - **yum install htop**
- **-p --pid**=PID,PID...
  - Show only the given PIDs

Linux

mands!

atop:

- **Usage**
  - atop is a system and process monitor for Linux systems that provides more detailed information than traditional tools like top
- **Features**
  - Displays CPU, memory, disk, and network utilization along with process-specific information, it also supports logging and replaying of system activity for analysis
- **Installation**
  - Installable via package managers (apt, yum) on Linux systems
    - **sudo yum install atop**
- **Example**
  - To start atop with all fields displayed
    - **atop -a**

iftop:

- **Usage:**
  - iftop is a command-line tool to monitor network bandwidth usage in real-time
- **Features**
  - Shows a list of network connections with detailed information about bandwidth usage on individual connections/interfaces. Useful for monitoring network traffic and identifying bandwidth
- **Installation**
  - Available in most Linux distributions. Install using package managers (apt, yum)
- **Example**
  - To view a summary of all available options and commands within iftop
    - **iftop -h**
  - If you want to monitor a specific interface, specify it with the -i option
    - **sudo iftop -i eth0**

nload:

- **Usage**
  - nload is another command-line tool for monitoring network traffic and bandwidth usage
- **Features**
  - Provides a visual representation of incoming and outgoing traffic on network interfaces using a graph. It also displays detailed statistics such as total data transferred and current transfer rates
- **Installation**
  - Installable via package managers (apt, yum) on Linux systems
    - **sudo yum install nload**
- **Example**
  - **-m :** Show multiple devices at a time; do not show the traffic graphs.
  - If you want to monitor a specific interface, specify it with the -i option
    - **sudo nload -i eth0**

ng!

System Administration Command - CronJobs

What is cronjobs

- Known as cron, is a job scheduler
- Used to run jobs periodically at fixed times, dates, or intervals
- A daemon process runs as a background process
- Performs specified operations at a predefined time or occurrence of an event without the intervention of the user
- Used to automate repeated tasks
- It starts automatically from **/etc/init.d** on entering the multi-user run levels

System Administration Command - CronJobs

- **Using crontab**
  - Crontab is a list of commands to execute the scheduled tasks at a specific time
  - Allows adding, removing and modifying the scheduled tasks
  - It has 6 fields, where the first 5 represent the time to run the task and the last one is for the command
    - **Minute** (value between 0-59)
    - **Hour** (value between 0-23)
    - **Day of Month** (value between 1-31)
    - **The month of the year** (value between 1-12 or Jan-Dec)
    - **Day of the week** (value between 0-6 or Sun-Sat)
    - Command

CronJobs(crontab)

Step: 01 Understanding cronjob syntax

Cronjobs use a specific syntax to schedule tasks. The syntax consists of five fields followed by the command to execute:

```
* * * * * command_to_execute
- - - - -
| | | | |
| | | | +---- Day of the week (0 - 7) (Sunday = 0 or 7)
| | | +------ Month (1 - 12)
| | +-------- Day of the month (1 - 31)
| +---------- Hour (0 - 23)
+------------ Minute (0 - 59)
```

Linux

Step 2: Viewing Existing Cron Jobs

- Command
  - **crontab -l**
- Lists all cronjobs for the current user

CronJobs(crontab)

Step 3: Creating a Simple CronJob

- Open the crontab editor:
  - **crontab -e**
- Add a cron job to run a script every day at 1:00 AM:
  - **0 1 * * * /path/to/your/script.sh**
- Replace /path/to/your/script.sh with the actual path to your script.
- **Save and exit the editor**

Step 4: Verifying the CronJob

- List the cron jobs again to verify:
  - **crontab -l**
- Ensure that your new cron job is listed

System-Wide Cron Jobs

Step	Command/Action
Open system-wide cron configuration	sudo nano /etc/crontab
Add a system-wide cron job	0 2 * * * root /path/to/your/script.sh
Save and exit	Save file and exit editor
Verify the changes	cat /etc/crontab

Step 9: Using the Cron Directories

Directory	Purpose	Example Command
/etc/cron.daily/	Scripts to run daily	sudo cp /path/to/your/script.sh /etc/cron.daily/
/etc/cron.weekly/	Scripts to run weekly	sudo cp /path/to/your/script.sh /etc/cron.weekly/
/etc/cron.monthly/	Scripts to run monthly	sudo cp /path/to/your/script.sh /etc/cron.monthly/
/etc/cron.hourly/	Scripts to run hourly	sudo cp /path/to/your/script.sh /etc/cron.hourly/

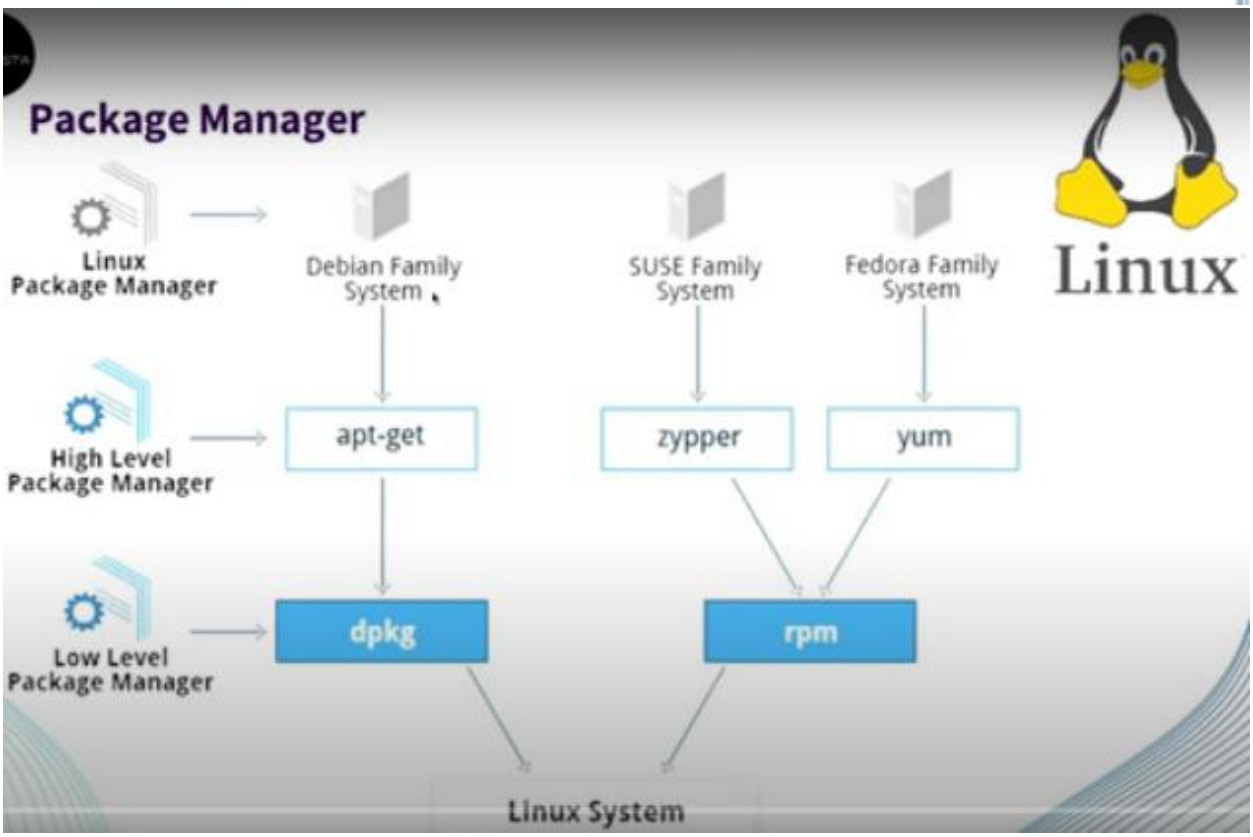
What we will learn today

Package Manager

Ubuntu Repo Management

Package Manager

- A collection of software tools that automates
  - The process of installing
  - Upgrading
  - Configuring, and removing computer programs for a computer in a consistent manner
- A package manager deals with packages, distributions of software and data in archive files



ckage Managers

- **apt:** Advanced Package Tool (Ubuntu, Debian)
- **yum:** Yellowdog Updater, Modified (CentOS)
- **rpm:** Red Hat Package Manager (RehHat)
- **homebrew:** package manager for MacOS
- **snap**: developed by Canonical for operating systems that use the Linux kernel

Package Manager

- **apt:** Advanced package tool
- **Yum:** A primary tool for getting, installing, deleting, querying, and otherwise managing Red Hat Enterprise Linux RPM software packages from official Red Hat software repositories
- **dpkg:** Install, remove and manage Debian packages
- **brew:** Free macOS package manager that allows you to install, update, or remove software by running commands in the terminal
- **snap:** Snaps are programs that are packaged with all their dependencies to run on all major Linux distros

Package Manager

- Upgrade all installed packages:
  - sudo apt upgrade
- Install a package:
  - sudo apt install package_name
- Remove a package:
  - sudo apt remove package_name
- Search for a package
  - apt search package_name
- Show package information:
  - apt show package_name
- Clean up unused packages:
  - sudo apt autoremove

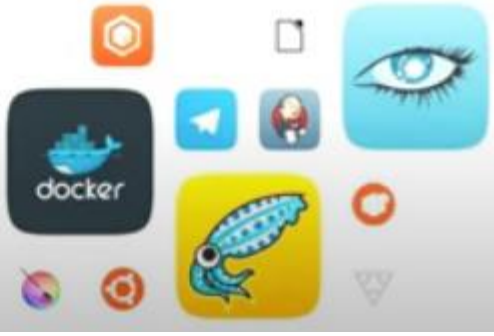
Snap Package Management

Snap is a package management system developed by Canonical that allows for easy installation and management of software packages in a containerized manner
Below are the key commands for managing Snap packages

Installing Snapd

- Update package list
  - sudo apt update
- Install Snap package manager
  - sudo apt install snapd

snapcraft



Key Features of Snap:

- **Universal Packages:** Snaps are designed to be universal packages that bundle the application and its dependencies together. This means snaps are independent of the underlying Linux distribution and its package manager
- **Rollback:** Snap supports rollback functionality, allowing users to revert to a previous version of an application if an update causes issues
- **Easy Installation:** Snaps can be installed with a single command, making it easy for users to discover and install applications from the Snap Store or other sources
- **Dependency Management:** Snaps include their dependencies, which are isolated from the rest of the system. This eliminates potential conflicts with other applications or system libraries

Summarizing the commands related to installing, managing, and viewing information about snaps using the snap command-line tool:

Command	Description
sudo snap install <snap_name>	Install a snap from the Snap Store or a local file
snap list	List all installed snaps on the system
sudo snap refresh <snap_name>	Update a snap to the latest version
sudo snap remove <snap_name>	Remove a snap from the system
snap find <search_term>	Search for snaps in the Snap Store using a search term
snap info <snap_name>	Display detailed information about a specific snap

Understanding APT Repositories

- APT (Advanced Package Tool) is a powerful command-line tool used to handle the installation and removal of software on Debian-based systems, such as Ubuntu

- It works with software repositories, which are collections of software packages, along with metadata used by APT tools to perform package management functions

APT Repositories:

- Collections of packages that are stored on remote servers and can be accessed over the internet. These repositories contain software applications, libraries, and other components that can be installed on your system

Repository Types:

- **Official Repositories:** Maintained by the Ubuntu project and contain software that is tested and integrated with the distribution
- **Personal Package Archives (PPAs):** Repositories provided by individual developers or teams to distribute software that may not be available in the official repositories
- **Third-Party Repositories:** Provided by external organizations or vendors to distribute proprietary or specialized software

Repository Files:

- Configuration files that list the repositories from which APT retrieves packages

  These files are:

  - **/etc/apt/sources.list:** The main configuration file for repositories
  - **/etc/apt/sources.list.d/:** A directory where additional repository files can be placed

Adding Repository

- **Adding a PPA Repository**
  - sudo add-apt-repository ppa:deadsnakes/ppa
  - sudo apt update
- **Manually Adding a Repository**
  - Open the Sources List File:
  - **sudo nano /etc/apt/sources.list**
- Add the Repository Line:
  - **deb http://archive.ubuntu.com/ubuntu/ bionic main restricted**
  - **Save and Close the File.**
- **Adding a GPG Key**
- To ensure the authenticity and integrity of the packages, add the repository's GPG key.
  - **wget -qO - https://dl.google.com/linux/linux_signing_key.pub | sudo apt-key add -**
- Updating the Package List
- After adding a repository or GPG key, update the package list to include packages from the new repository:
- Update Packages
  - **sudo apt update**

What we will learn today

Overview of fdisk

Listing Existing Partitions

Creating a New Partition

Modifying a Partition

Objective

Summary

Linux

fdisk

fdisk is a powerful command-line tool used to manage disk partitions. Here's a more detailed guide to using fdisk effectively:

List All Partitions

List partitions on all disks:

- **sudo fdisk -l**

This command will display all the partitions on all the available disks, including details such as device names, sizes, and types.

Listing Existing Partitions

Step	Command/Action	Description
Open fdisk	sudo fdisk /dev/sdX	Replace X with the actual disk identifier
List partitions	p	Displays the partition table of the disk

```
ubuntu@babarzahoor:~$ sudo fdisk /dev/sda

Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

This disk is currently in use - repartitioning is probably a bad idea.
It's recommended to umount all file systems, and swapoff all swap
partitions on this disk.


Command (m for help):
```

Creating a New Partition

Step	Command/Action	Description
Open fdisk	sudo fdisk /dev/sdX	Replace X with the actual disk identifier
Create a new partition	n	Starts the process of creating a new partition
Select partition type	p for primary, e for extended	Choose between primary and extended partition
Specify partition number	Enter a number (1-4 for primary)	Specifies the partition number
Specify the first sector	Default (press Enter) or enter a value	Defines the starting point of the partition
Specify the last sector	Default (press Enter) or enter a value	Defines the ending point or size of the partition

What we will learn today

Init
Run level
Systemd

Linux

temD!

Init

- Stands for Initialization
- The first process started during the booting of the computer system
- daemon process that continues running until the system is shutdown
- A direct or indirect ancestor of all other processes
- automatically adopts all orphaned processes
- PID or Process ID is 1
- create processes from a script stored in /etc/inittab
  - configuration file used by initialization system
  - last step of kernel boot sequence

Init

- Init script initializes the service
- responsible for initializing the system
- init scripts are also called rc scripts (run command scripts)
- Also used in UNIX

Init - Services

- Start a service
  - service [service-name] start
- Stop a service
  - service [service-name] stop
- restart a service
  - service [service-name] restart
- reload a service
  - service [service-name] reload
- service status
  - service [service-name] status

temD!

Init - Services

- Restart service if already running
  - service [service-name] condrestart
- Enable service at startup
  - chkconfig [service-name] on
- Disable service at startup
  - chkconfig [service-name] off
- Check if service is enabled at startup
  - chkconfig [service-name]
- Create new service file or modify configuration
  - chkconfig [service-name] add

temD!

Run-levels

- State of init
- Group of processes are defined to start at the startup of OS
- Mode of operation in the computer operating system
- Each run level has a certain number of services started or stopped
- **Seven Run levels exist** ($ runlevel digit or $ init digit)
  - 0: Shuts down the system
  - 1: single-user mode
  - 2: multi-user mode without networking
  - 3: multi-user mode with networking (with CLI)
  - 4: user-definable
  - 5: multi-user mode with networking (with GUI)
  - 6: reboots the system to restart it

temD!

Systemd

- Systemd is the new init framework
- System and service manager for Linux operating systems
- New distros are moving to systemd
- Managing services with systemd
  - **systemctl:** Control system and services
  - **journalctl:** Manage journal, systemd's logging system
  - **hostnamectl:** Control hostname
  - **localectl:** Configure locale and keyboard layout
  - **timedatectl:** Set time and date
  - **systemd-cgls**: Shows cgroup contents
  - **systemadm:** Front-end for systemctl

emD!

Systemd - Services

- See all services (running or not)
  - **systemctl list-units --type service --all**
- Start a service
  - **systemctl start [service-name]**
- Stop a service
  - **systemctl stop [service-name]**
- Restart a service
  - **systemctl restart [service-name]**
- Reload a service
  - **systemctl reload [service-name]**

temD!

Systemd - Services

- This command instructs systemd to reload its configuration files, including unit files and security profiles, without restarting the system or services
  - **systemctl daemon-reload**
- This command shuts down the system immediately.
  - **systemctl poweroff**
- Halt stops the OS and then we can proceed with powering down the device after halting
  - **systemctl halt**
- It instructs systemd to restart the entire system
  - **systemctl reboot**

What we will learn today

Log Directory

Syslog

Journalctl

Kernel Log

Log Directory

- **/var/log**
  - Special directory for storing logs
  - Logs for Operating systems, Services, and various applications
  - In debian-based systems, /var/log/syslog stores global activity data, including startup messages. Same is stored in /var/log/messages in RedHat or CentOS
  - In Debian-based systems, /var/log/auth.log store all security-related events such as logins, root user actions, and output from pluggable authentication modules (PAM). Red Hat and CentOS use /var/log/secure

Log Directory

- **/var/log/kern.log** stores kernel events, errors, and warning logs, which are particularly helpful for troubleshooting custom kernels
- **/var/log/cron** stores information about scheduled tasks (cron jobs) Use this data to verify that your cron jobs are running successfully
- Some applications also write log files in this directory. For example, the Apache web server writes logs to the /var/log/apache2 directory, while MySQL writes logs to the /var/log/mysql directory

Syslog

The standard for creating and transmitting logs

- **Syslog service:** receives and processes Syslog messages
  - Listens to events by creating a socket located at /dev/log where applications write
  - It can write messages to a local file or forward messages to a remote server
  - Different implementations including rsyslogd and Syslog-ng
  - Syslog Protocol RFC5424: transport protocol that specifies how to transmit logs over a network
  - Also defines the format and how messages are structured
  - uses port 514 for plaintext messages and 6514 for encrypted messages
- **Syslog messages:** any log formatted in Syslog message format
  - consists of a standardized header and message containing the log's content
  - A standardized header includes a timestamp, name of the application, location where the message originated, and its priority

ctl, and Kernel Logs

Journalctl

- It is a utility for querying and displaying logs from journald.
- journald stores log data in binary format instead of plaintext, journalctl is the way to read log messages processed by journald
- for complete journalctl options, visit journalctl man page
- to show all journal enteries, mentioning the start and end time, use simple "journalctl" command without any options

lctl, and Kernel Logs

Journalctl

- for unit, or a service, using -u
  - **journalctl -u nginx.service**
- to follow or tail the logs, use -f
  - **journalctl -f**
  - **journalctl -u mysql.service -f**
- for output formats, user -o (jsonwill, json-pretty, verbose, cat, short, short-precise)
  - **journactl -u nginx.service -r -o json-pretty**

ctl, and Kernel Logs

Kernel Log

- **/var/log/dmesg**
  - system logs at boot time about kernel ring buffer
  - Shows information about hardware drivers, kernel information and status during bootup
  - It gets reset on every reboot
  - if having issues with something during bootup or hardware issue, dmesg is the best place to look at.
  - can view this log using 'dmesg' command
- **/var/log/kern.log**
  - logs kernel information and events on the system.
  - also logs dmesg output

Firewall

Firewall Definition:

- Network security device or software
- Monitors and controls incoming/outgoing traffic
- Establishes a barrier between trusted and untrusted networks

Firewall Rules:

- **Allow Rules:** Permit specified traffic
- **Deny Rules:** Block specified traffic
- **Default Deny Policy**: Deny all traffic by default, only allow specified traffic

Functions and Features:

- **Traffic Monitoring:**
  - Monitor network traffic for suspicious activity
- **Access Control:**
  - Restrict access based on user roles/IP addresses
- **Logging and Reporting:**
  - Provide logs/reports for auditing and analysis
- **VPN Support:**
  - Facilitate secure remote access

l Management

Firewall

- A network security system that monitors and controls the incoming and outgoing network traffic based on predefined rules
- It typically establishes a barrier between a trusted network and an untrusted network
- A firewall can be a software firewall, hardware firewall or both
- Ubuntu has its own firewall known as the uncomplicated firewall (UFW)
- simplified process of configuring a firewall
- install ufw using:
  - **sudo apt install ufw**
- Default configuration can be found at **/etc/default/ufw**

Firewall - ufw

- Check Firewall Status
  - **sudo ufw status**

```
ubuntu@babarzahoor:~$ sudo ufw status
[sudo] password for ubuntu:
Status: inactive
```

Basic ufw Commands

- Enable Firewall
  - **sudo ufw enable**

```
ubuntu@babarzahoor:~$ sudo ufw enable
Firewall is active and enabled on system startup
```

- Disable Firewall
  - **sudo ufw disable**
- View the app list in ufw
  - **sudo ufw app list**

Linux

To enable ssh

- **Commands**
  - sudo ufw allow OpenSSH
  - sudo ufw allow ssh
  - sudo ufw allow 22

Allowing and Denying Connections

Action	Command	Description
Allow a port	sudo ufw allow 22	Allows incoming connections on port 22 (SSH)
Deny a port	sudo ufw deny 22	Denies incoming connections on port 22
Allow a specific IP	sudo ufw allow from 192.168.1.100	Allows connections from a specific IP address
Deny a specific IP	sudo ufw deny from 192.168.1.100	Denies connections from a specific IP address
Allow a port on a specific interface	sudo ufw allow in on eth0 to any port 80	Allows incoming connections on port 80 on eth0
Deny a port on a specific interface	sudo ufw deny in on eth0 to any port 80	Denies incoming connections on port 80 on eth0
Allow a range of ports	sudo ufw allow 1000:2000/tcp	Allows incoming TCP connections on ports 1000-2000
Deny a range of ports	sudo ufw deny 1000:2000/tcp	Denies incoming TCP connections on ports 1000-2000

Table 1: Introduction and Installation

Step	Action	Command/Instructions
1. Introduction to OpenSSH	Overview of OpenSSH	• **A suite for secure remote login and file transfer.**
2. Installing OpenSSH Server	For Ubuntu:	• **sudo apt update** • **sudo apt install openssh-server**
3. Configuring OpenSSH Server	Edit Configuration File	• **/etc/ssh/sshd_config**
	Change SSH Port	**Port 22**
	Disable Root Login	**PermitRootLogin no**
	Restart SSH Service	• **sudo systemctl restart sshd**

Table 2: Checking Status and Key Management

Step	Action	Command/Instructions
4. Checking SSH Server Status	Check if SSH server is running	• **sudo systemctl status sshd**
	Enable SSH on Startup	• **sudo systemctl enable sshd**
5. Generating SSH Keys	Generate SSH Key Pair	• **ssh-keygen -t rsa -b 4096**
6. Understanding SSH Key Files	Key Pair Components	• **Private Key: ~/.ssh/id_rsa** • **Public Key: ~/.ssh/id_rsa.pub**

Table 3: Key Usage and Security

Step	Action	Command/Instructions
7. Copying SSH Public Key	Use ssh-copy-id	• **ssh-copy-id username@remote_server_ip**
	Alternative Method	• **`cat ~/.ssh/id_rsa.pub**
8. Logging into Remote Server	Log in using SSH	• **ssh username@remote_server_ip**
	Use Different Port	• **ssh -p 2222 username@remote_server_ip**
9. Passwordless Login	Check for Passwordless Login	• **ssh username@remote_server_ip**

Table 3: Key Usage and Security

Step	Action	Command/Instructions
10. Managing SSH Keys	View Existing Keys	• **ls ~/.ssh/**
	Remove a Specific Key	• **ssh-keygen -R remote_server_ip**
	Add/Replace Public Keys	• **Edit ~/.ssh/authorized_keys**
11. Securing Your SSH Configuration	Change Default SSH Port	**Update Port 22 in sshd_config**
	Restrict SSH Access	**AllowUsers username**
	Disable Password Authentication	**PasswordAuthentication no**
	Restart SSH Service	**sudo systemctl restart sshd**