Linux over Windows

Cost-Effectiveness

- **Free and Open Source**: Linux does not require expensive licensing fees, making it a cost-effective choice for companies.
- **Lower Maintenance Costs**: Linux is stable and requires minimal maintenance, reducing operational expenses.

Performance and Efficiency

- **Better Resource Utilization**: Linux is lightweight and consumes fewer system resources compared to Windows.
- **High Scalability**: Linux efficiently scales from small embedded systems to enterprise data centers without performance degradation.

Security and Reliability

- **Less Vulnerable to Malware**: Linux has strong user privilege separation, making it more secure against viruses and malware.
- **Frequent and Transparent Updates**: Regular security patches ensure system stability without requiring frequent reboots.
- **High Stability**: Linux systems can run for years without crashes, ensuring better uptime and reliability.

Core components of a Linux Machine

```
+----------------------------------------------------+
| User Applications (Vim, Docker, Apache, etc.)     |
+----------------------------------------------------+
| Shell (Bash, Zsh, Fish, etc.)                     |  <-- Part of the OS
+----------------------------------------------------+
| System Libraries (glibc, libc, OpenSSL, etc.)     |  <-- Part of the OS
+----------------------------------------------------+
| System Utilities (ls, grep, systemctl, etc.)      |  <-- Part of the OS
+----------------------------------------------------+
| Linux Kernel (Process, Memory, FS, Network)       |  <-- Core of the OS
+----------------------------------------------------+
| Hardware (CPU, RAM, Disk, Network, Peripherals)   |
+----------------------------------------------------+


(a) Hardware Layer

🔹 The physical components of the computer (CPU, RAM, disk, network interfaces, etc.).
🔹 The OS interacts with hardware using device drivers.
(b) Kernel (Core of Linux OS)

🔹 The Linux Kernel is responsible for directly managing system resources, including:

    Process Management – Schedules processes and handles multitasking.

    Memory Management – Allocates and deallocates RAM efficiently.

    Device Drivers – Acts as an interface between software and hardware.

    File System Management – Manages how data is stored and retrieved.

    Network Management – Handles communication between systems.

(c) Shell (Command Line Interface - CLI)

🔹 A command interpreter that allows users to interact with the kernel.
🔹 Examples: Bash, Zsh, Fish, Dash, Ksh.
🔹 Converts user commands into system calls for the kernel.
(d) User Applications

🔹 End-user programs like web browsers, text editors, DevOps tools, etc.
🔹 Applications interact with the OS using system calls via the shell or GUI.
```

Linux Distributions

Here are some popular Linux distributions:

```
Ubuntu – One of the most beginner-friendly distros, widely used for personal and server use.
It has great community support.

CentOS (discontinued, replaced by AlmaLinux/Rocky Linux) – Previously a popular choice for
servers, based on Red Hat Enterprise Linux (RHEL).

Debian – A very stable and reliable distro, often used as a base for other distros like
Ubuntu.

Fedora – A cutting-edge distro that introduces new features before they reach RHEL.

Arch Linux – A lightweight, rolling-release distro for advanced users who like
customization.

Kali Linux – Designed for cybersecurity and penetration testing.

Alpine Linux – A lightweight, security-focused distro often used in containers.
```

Useful References:

- Linux Kernel Source code:
  http://git.kernel.org/
- Mirror of Linux Kernel on GitHub:
  http://github.com/torvalds/linux

Setup Linux Environment on Windows and MacOS

There are multiple ways to setup a Linux environment on a Windows or Mac machines such as `cloud vm`, `wsl2`, `virtualbox`, `Hyperkit` e.t.c.,. However what I would recommend is using a container as a Linux environment.

Just install Docker desktop, run the below command and create linux container of any distribution without worrying about the cost and connectivity issues.

Docker Command to Run Ubuntu Linux Container in windows host (Persistent & Long-Term)

- Create a folder with name `ubuntu-data` in your downloads folder.
- Then run the below command in `poweshell` updating your `username`.

```
docker run -dit `
  --name ubuntu-container `
  --hostname ubuntu-dev `
  --restart unless-stopped `
  --cpus="2" `
  --memory="4g" `
  --mount type=bind,source="C:/Users/Monica Korla/Downloads/ubuntu-container",target=/data `
  -v /var/run/docker.sock:/var/run/docker.sock `
  -p 2222:22 `
  -p 8080:80 `
  --env TZ=Asia/Kolkata `
  --env LANG=en_US.UTF-8 `
  ubuntu:latest /bin/bash
```

Docker Command to Run Ubuntu Linux Container in mac or linux host (Persistent & Long-Term)

```
docker run -dit \
  --name ubuntu-container \
  --hostname ubuntu-dev \
  --restart unless-stopped \
  --cpus="2" \
  --memory="4g" \
  --mount type=bind,source=/tmp/ubuntu-data,target=/data \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -p 2222:22 \
  -p 8080:80 \
  --env TZ=Asia/Kolkata \
  --env LANG=en_US.UTF-8 \
  ubuntu:latest /bin/bash
```

Explanation of Each Parameter

Parameter	Description
`-dit`	Runs the container in **detached (-d)**, **interactive (-i)**, and **terminal (-t)** mode.
`--name ubuntu-container`	Assigns a name to the container for easy management.
`--hostname ubuntu-dev`	Sets the container's hostname.
`--restart unless-stopped`	Ensures the container restarts automatically unless manually stopped.
`--cpus="2"`	Limits the container to **2 CPU cores**.
`--memory="4g"`	Allocates **4GB RAM** to the container.
`--mount type=bind,source=C:/ubuntu-data,target=/data`	**Mounts a folder** from Windows into the container to persist data.
`-v /var/run/docker.sock:/var/run/docker.sock`	Allows running Docker commands inside the container (optional).
`-p 2222:22`	Maps port **2222** on the host to **22** (SSH) inside the container.
`-p 8080:80`	Maps port **8080** on the host to **80** (for web services).
`--env TZ=Asia/Kolkata`	Sets the **timezone** (modify based on your location).
`--env LANG=en_US.UTF-8`	Sets the **language** settings inside the container.
`ubuntu:latest /bin/bash`	Uses the latest **Ubuntu** image and runs Bash shell.

Package Managers in Linux

📌 What is a Package Manager?

A **package manager** is a tool that automates the process of installing, updating, configuring, and removing software in a Linux system. It ensures that software and its dependencies are managed efficiently.

🔍 How Does a Package Manager Work?

1. **Repositories (Repos):**
    - A package manager fetches software from **official repositories (online storage of packages).**
    - Example: Ubuntu gets packages from `archive.ubuntu.com`.
2. **Installing Software:**
    - When you install software, the package manager:
      ✅ Downloads the package from the repository.
      ✅ Resolves dependencies (installs additional required software).
      ✅ Installs and configures the software automatically.
3. **Updating Software:**
    - A single command updates all installed packages to the latest version.
4. **Removing Software:**
    - The package manager also **removes** software cleanly without leaving unnecessary files.

📦 Popular Package Managers in Linux

Linux Distro	Package Manager	Command Example
Ubuntu, Debian	`apt` (Advanced Package Tool)	`sudo apt install nginx`
Fedora, RHEL, CentOS	`dnf` (or `yum` for older versions)	`sudo dnf install nginx`
Arch Linux	`pacman`	`sudo pacman -S nginx`
OpenSUSE	`zypper`	`sudo zypper install nginx`

🌍 How Package Managers Fetch Software from Repositories

A **repository** is a server that stores software packages. When a package manager installs software:

1. It **checks the repository list** (e.g., `/etc/apt/sources.list` in Ubuntu).
2. It **downloads the package** and its dependencies.
3. It **installs and configures the software** automatically.

📁 Example of an Ubuntu Repository Entry

```
Types: deb
URIs: http://ports.ubuntu.com/ubuntu-ports/
Suites: noble noble-updates noble-backports noble-security
Components: main universe restricted multiverse
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg
```

🔄 Why Should You Run `apt update` After Installing Ubuntu?

When you install Ubuntu, the packages included in the ISO image might be outdated. Running:

```
apt install sudo
sudo apt update
```

✅ Updates the package list from repositories.

Then, to install the latest versions of packages, run:

```
sudo apt upgrade -y
```

🛠️ Essential Package Manager Commands

APT (Debian, Ubuntu)

```
sudo apt update          # Update package lists
sudo apt upgrade -y      # Upgrade installed packages
sudo apt install nginx   # Install a package
sudo apt remove nginx    # Remove a package
sudo apt autoremove      # Remove unused dependencies
sudo apt search nginx    # Search for a package
```

DNF (Fedora, RHEL, CentOS)

```
sudo dnf check-update    # Check for updates
sudo dnf update          # Update all packages
sudo dnf install nginx   # Install a package
sudo dnf remove nginx    # Remove a package
```

Pacman (Arch Linux)

```
sudo pacman -Syu         # Sync and update all packages
sudo pacman -S nginx     # Install a package
sudo pacman -R nginx     # Remove a package
```

Zypper (OpenSUSE)

```
sudo zypper refresh     # Refresh package list
sudo zypper update      # Update all packages
sudo zypper install nginx  # Install a package
sudo zypper remove nginx   # Remove a package
```

🚀 Best Practices for Using Package Managers

- ✅ **Always update your package list before installing software:**

  ```
  sudo apt update && sudo apt upgrade -y
  ```

- ✅ **Use `autoremove` to clean up unused dependencies:**

  ```
  sudo apt autoremove
  ```

- ✅ **Enable automatic security updates (Ubuntu):**

  ```
  sudo apt install unattended-upgrades
  sudo dpkg-reconfigure unattended-upgrades
  ```

This document provides a solid foundation for understanding package managers in Linux! 🚀

Understanding the Folder Structure

Explanation of System Directories

Symbolic Links (Less Significant)

Directory	Description
`/sbin -> /usr/sbin`	System binaries for administrative commands (linked to `/usr/sbin`).
`/bin -> /usr/bin`	Essential user binaries (linked to `/usr/bin`).
`/lib -> /usr/lib`	Shared libraries and kernel modules (linked to `/usr/lib`).

Important System Directories

Directory	Description
`/boot`	Stores files needed for booting the system (not relevant in containers).
`/usr`	Contains most user-installed applications and libraries.
`/var`	Stores logs, caches, and temporary files that change frequently.
`/etc`	Stores system configuration files.

User & Application-Specific Directories

Directory	Description
`/home`	Default location for user home directories.
`/opt`	Used for installing optional third-party software.
`/srv`	Holds data for services like web servers (rarely used in containers).
`/root`	Home directory for the root user.

Temporary & Volatile Directories

Directory	Description
`/tmp`	Temporary files (cleared on reboot).
`/run`	Holds runtime data for processes.
`/proc`	Virtual filesystem for process and system information.
`/sys`	Virtual filesystem for hardware and kernel information.

Directory	Description
`/dev`	Contains device files (e.g., `/dev/null`, `/dev/sda`).

Mount Points

Directory	Description
`/mnt`	Temporary mount point for external filesystems.
`/media`	Mount point for removable media (USB, CDs).
`/data`	Likely your **mounted volume** from Windows (`C:/ubuntu-data`).

User Management in Linux

Introduction to User Management in Linux

Linux is a multi-user operating system, meaning multiple users can operate on a system simultaneously. Proper user management ensures security, controlled access, and system integrity.

Key files involved in user management:

- `/etc/passwd` – Stores user account details.
- `/etc/shadow` – Stores encrypted user passwords.
- `/etc/group` – Stores group information.
- `/etc/gshadow` – Stores secure group details.

Creating Users in Linux

To create a new user in Linux, use:

`useradd` Command (For most Linux distributions)

```
useradd username
```

This creates a user without a home directory.

To create a user with a home directory:

```
useradd -m username
```

To specify a shell:

```
useradd -s /bin/bash username
```

`adduser` Command (For Debian-based systems)

```
adduser username
```

This is an interactive command that asks for a password and additional details.

Managing User Passwords

To set or change a user’s password:

```
passwd username
```

Enforcing Password Policies

- **Password expiration**: Set password expiry days

  ```
  chage -M 90 username
  ```

- **Lock a user account**

  ```
  passwd -l username
  ```

- **Unlock a user account**

  ```
  passwd -u username
  ```

Modifying Users

Modify an existing user with `usermod`:

- Change the username:

  ```
  usermod -l new_username old_username
  ```

- Change the home directory:

  ```
  usermod -d /new/home/directory -m username
  ```

- Change the default shell:

  ```
  usermod -s /bin/zsh username
  ```

Deleting Users

To remove a user but keep their home directory:

```
userdel username
```

To remove a user and their home directory:

```
userdel -r username
```

Working with Groups

Creating Groups

```
groupadd groupname
```

Adding Users to Groups

```
usermod -aG groupname username
```

Viewing Group Memberships

```
groups username
```

Changing Primary Group

```
usermod -g new_primary_group username
```

Sudo Access and Privilege Escalation

Adding a User to Sudo Group

On Debian-based systems:

```
usermod -aG sudo username
```

On RHEL-based systems:

```
usermod -aG wheel username
```

Granting Specific Commands with Sudo

Edit the sudoers file:

```
visudo
```

Then add:

```
username ALL=(ALL) NOPASSWD: /path/to/command
```

File management in Linux

File and Directory Management

1. `ls` – Lists files and directories in the current location.
2. `cd /path/to/directory` – Changes the working directory.
3. `pwd` – Prints the current working directory.
4. `mkdir new_folder` – Creates a new directory.
5. `rmdir empty_folder` – Removes an empty directory.
6. `rm file.txt` – Deletes a file.
7. `rm -r folder` – Deletes a folder and its contents.
8. `cp file1.txt file2.txt` – Copies a file.
9. `cp -r dir1 dir2` – Copies a directory recursively.
10. `mv old_name new_name` – Moves or renames a file or directory.

File Viewing and Editing

11. `cat file.txt` – Displays file content.
12. `tac file.txt` – Displays file content in reverse order.
13. `less file.txt` – Opens a file for viewing with scrolling support.
14. `more file.txt` – Similar to `less`, but only moves forward.
15. `head -n 10 file.txt` – Displays the first 10 lines of a file.
16. `tail -n 10 file.txt` – Displays the last 10 lines of a file.
17. `nano file.txt` – Opens a simple text editor.
18. `vi file.txt` – Opens a powerful text editor.
19. `echo 'Hello' > file.txt` – Writes text to a file, overwriting existing content.
20. `echo 'Hello' >> file.txt` – Appends text to a file without overwriting.

VI Editor Shortcuts

Modes in VI Editor

- **Normal Mode** (default) – Used for navigation and command execution.
- **Insert Mode** – Used for text editing (press `i` to enter, `Esc` to exit).
- **Command Mode** – Used for saving, quitting, and searching (press `:` in Normal mode).

Basic Navigation

- `h` – Move **left**
- `l` – Move **right**
- `j` – Move **down**
- `k` – Move **up**
- `0` – Move to the **beginning** of the line
- `^` – Move to the **first non-blank** character of the line
- `$` – Move to the **end** of the line
- `w` – Move to the **next word**
- `b` – Move to the **previous word**
- `gg` – Move to the **start** of the file
- `G` – Move to the **end** of the file
- `:n` – Move to **line number** `n`

Insert Mode Shortcuts

- `i` – Insert before cursor
- `I` – Insert at the beginning of the line
- `a` – Append after cursor
- `A` – Append at the end of the line
- `o` – Open a new line below
- `O` – Open a new line above
- `Esc` – Exit insert mode

Editing Text

- `x` – Delete a **character**
- `X` – Delete a **character before cursor**
- `dw` – Delete a **word**
- `dd` – Delete a **line**
- `d$` – Delete from **cursor to end of line**
- `d0` – Delete from **cursor to beginning of line**
- `D` – Delete from **cursor to end of line**
- `u` – **Undo** last action
- `Ctrl + r` – **Redo** an undone change
- `yy` – Copy (yank) a **line**
- `yw` – Copy (yank) a **word**
- `p` – Paste **after** the cursor
- `P` – Paste **before** the cursor

Search and Replace

- `/pattern` – Search **forward** for a pattern
- `?pattern` – Search **backward** for a pattern
- `n` – Repeat last search **forward**
- `N` – Repeat last search **backward**
- `:%s/old/new/g` – Replace **all occurrences** of "old" with "new"
- `:s/old/new/g` – Replace **all occurrences** in the current line

Working with Multiple Files

- `:e filename` – Open a **new file**
- `:w` – Save file
- `:wq` – Save and exit
- `:q!` – Quit **without saving**
- `:split filename` – Split screen **horizontally** and open another file
- `:vsplit filename` – Split screen **vertically**
- `Ctrl + w + w` – Switch between split screens

File Permissions Management in Linux

Introduction to File Permissions

Linux file permissions determine who can read, write, or execute files and directories. Each file and directory has three levels of permission:

- **Owner (User)**: The creator of the file.
- **Group**: Users belonging to the assigned group.
- **Others**: All other users on the system.

Permissions are represented as:

- **Read (`r` or `4`)** – View file contents.
- **Write (`w` or `2`)** – Modify file contents.
- **Execute (`x` or `1`)** – Run scripts or programs.

To check file permissions, use:

```
ls -l filename
```

Output example:

```
-rwxr--r-- 1 user group 1234 Mar 28 10:00 myfile.sh
```

Changing Permissions with `chmod`

Using Symbolic Mode

Modify permissions using symbols:

- Add (`+`), remove (`-`), or set (`=`) permissions.

Examples:

```
chmod u+x filename  # Add execute for user
chmod g-w filename  # Remove write for group
chmod o=r filename  # Set read-only for others
chmod u=rwx,g=rx,o= filename  # Set full access for user, read/execute for group, and no access for others
```

Using Numeric (Octal) Mode

Each permission has a value:

- Read (`4`), Write (`2`), Execute (`1`).

Examples:

```
chmod 755 filename  # User (rwx), Group (r-x), Others (r-x)
chmod 644 filename  # User (rw-), Group (r--), Others (r--)
chmod 700 filename  # User (rwx), No access for others
```

Changing Ownership with `chown`

Modify file owner and group:

```
chown newuser filename  # Change owner
chown newuser:newgroup filename  # Change owner and group
chown :newgroup filename  # Change only group
```

Recursively change ownership:

```
chown -R newuser:newgroup directory/
```

Changing Group Ownership with `chgrp`

```
chgrp newgroup filename  # Change group
chgrp -R newgroup directory/  # Change group recursively
```

Special Permissions

SetUID (`s` on user execute bit)

Allows users to run a file with the file owner's permissions.

```
chmod u+s filename
```

Example: `/usr/bin/passwd` allows users to change their passwords.

SetGID (`s` on group execute bit)

Files: Users run the file with the group's permissions.
Directories: Files created inside inherit the group.

```
chmod g+s filename  # Set on file
chmod g+s directory/  # Set on directory
```

Sticky Bit (`t` on others execute bit)

Used on directories to allow only the owner to delete their files.

```
chmod +t directory/
```

Example: `/tmp` directory.

Default Permissions: `umask`

`umask` defines default permissions for new files and directories.
Check current umask:

```
umask
```

Set a new umask:

```
umask 022  # Default: 755 for directories, 644 for files
```

Conclusion

Understanding file permissions is essential for system security and proper file management. Using `chmod`, `chown`, and `chgrp`, you can control access to files and directories efficiently.

Networking Commands

1. `ping google.com` – Checks connectivity to a remote server.
2. `ifconfig` – Displays network interfaces (deprecated, use `ip`).
3. `ip a` – Shows IP addresses of network interfaces.
4. `netstat -tulnp` – Displays open network connections.
5. `curl https://example.com` – Fetches a webpage's content.
6. `wget https://example.com/file.zip` – Downloads a file from the internet.

Process Management in Linux

Introduction to Process Management

A process is an instance of a running program. Linux provides multiple utilities to monitor, manage, and control processes effectively. Each process has a unique **Process ID (PID)** and belongs to a parent process.

Index of Commands Covered

Viewing Processes

- `ps aux` – View all running processes
- `ps -u username` – View processes for a specific user
- `ps -C processname` – Show a process by name
- `pgrep processname` – Find a process by name and return its PID
- `pidof processname` – Find the PID of a running program

Managing Processes

- `kill PID` – Terminate a process by PID
- `pkill processname` – Terminate a process by name
- `kill -9 PID` – Force kill a process
- `pkill -9 processname` – Kill all instances of a process
- `kill -STOP PID` – Stop a running process
- `kill -CONT PID` – Resume a stopped process
- `renice -n 10 -p PID` – Lower priority of a process
- `renice -n -5 -p PID` – Increase priority of a process (requires root)

Background & Foreground Processes

- `command &` – Run a command in the background
- `jobs` – List background jobs
- `fg %jobnumber` – Bring a job to the foreground
- `Ctrl + Z` – Suspend a running process
- `bg %jobnumber` – Resume a suspended process in the background

Monitoring System Processes

- `top` – Interactive process viewer
- `htop` – User-friendly process viewer (requires installation)
- `nice -n 10 command` – Run a command with a specific priority
- `renice -n -5 -p PID` – Change priority of an existing process

Daemon Process Management

- `systemctl list-units --type=service` – List all system daemons
- `systemctl start service-name` – Start a daemon/service
- `systemctl stop service-name` – Stop a daemon/service
- `systemctl enable service-name` – Enable a service at startup

Viewing Process Details

Using `ps`

Show processes for a specific user:

```
ps -u username
```

Show a process by name:

```
ps -C processname
```

Using `pgrep`

Find a process by name and return its PID:

```
pgrep processname
```

Using `pidof`

Find the PID of a running program:

```
pidof processname
```

Managing Processes

Killing Processes

To terminate a process by PID:

```
kill PID
```

To terminate using process name:

```
pkill processname
```

Force kill a process:

```
kill -9 PID
```

Kill all instances of a process:

```
pkill -9 processname
```

Stopping & Resuming Processes

Stop a running process:

```
kill -STOP PID
```

Resume a stopped process:

```
kill -CONT PID
```

Changing Process Priority

View process priorities:

```
top  # Look at the NI column
```

Change priority of a running process:

```
renice -n 10 -p PID  # Lower priority (positive values)
renice -n -5 -p PID  # Higher priority (negative values, root required)
```

Running Processes in the Background

Run a command in the background:

```
command &
```

List background jobs:

```
jobs
```

Bring a job to the foreground:

```
fg %jobnumber
```

Send a running process to the background:

```
Ctrl + Z  # Suspend process
bg %jobnumber  # Resume in background
```

Monitoring System Processes

Using `top`

Interactive process viewer:

- Press `k` and enter a PID to kill a process.
- Press `r` to renice a process.
- Press `q` to quit.

Using `htop`

A user-friendly alternative to `top`:

```
htop
```

Allows mouse-based interaction for process management.

Using `nice` & `renice`

Run a command with a specific priority:

```
nice -n 10 command
```

Change the priority of an existing process:

```
renice -n -5 -p PID
```

Daemon Processes

Daemon processes run in the background without user intervention.
List all system daemons:

```
systemctl list-units --type=service
```

Start a daemon:

```
systemctl start service-name
```

Stop a daemon:

```
systemctl stop service-name
```

Enable a service at startup:

```
systemctl enable service-name
```

Conclusion

Process management is crucial for system performance and stability. By using tools like `ps`, `top`, `htop`, `kill`, and `nice`, you can efficiently control and monitor Linux processes.

Linux System Monitoring

Introduction to System Monitoring

Monitoring system resources is essential to ensure optimal performance, detect issues, and troubleshoot problems in Linux. Various tools allow us to monitor CPU, memory, disk usage, network activity, and running processes.

Index of Commands Covered

CPU and Memory Monitoring

- `top` – Real-time system monitoring
- `htop` – Interactive process viewer (requires installation)
- `vmstat` – Report system performance statistics
- `free -m` – Show memory usage

Disk Monitoring

- `df -h` – Check disk space usage
- `du -sh /path` – Show disk usage of a specific directory
- `iostat` – Display CPU and disk I/O statistics

Network Monitoring

- `ifconfig` – Show network interfaces (deprecated, use `ip a`)
- `ip a` – Show network interface details
- `netstat -tulnp` – Show active connections and listening ports
- `ss -tulnp` – Alternative to `netstat` for socket statistics
- `ping hostname` – Test network connectivity
- `traceroute hostname` – Show network path to a host
- `nslookup domain` – Get DNS resolution details

Log Monitoring

- `tail -f /var/log/syslog` – Live monitoring of system logs
- `journalctl -f` – Live system logs for systemd-based distros
- `dmesg | tail` – View kernel logs

CPU and Memory Monitoring

Using `top`

To view real-time CPU and memory usage:

```
top
```

Press `q` to quit.

Using `htop`

A user-friendly alternative:

```
htop
```

Use arrow keys to navigate and `F9` to kill processes.

Using `vmstat`

To check CPU, memory, and I/O stats:

```
vmstat 1 5  # Update every 1 sec, show 5 updates
```

Checking Memory Usage

```
free -m
```

Shows free and used memory in megabytes.

Disk Monitoring

Using `df`

Check available disk space:

```
df -h
```

Using `du`

Find the size of a directory:

```
du -sh /var/log
```

Using `iostat`

Check disk and CPU usage:

```
iostat
```

Network Monitoring

Checking Network Interfaces

```
ip a  # Show IP addresses and interfaces
```

Viewing Open Ports and Connections

```
netstat -tulnp  # Show listening ports
ss -tulnp  # Alternative to netstat
```

Testing Connectivity

```
ping google.com  # Test internet connection
traceroute google.com  # Trace the path to Google
```

Checking DNS Resolution

```
nslookup example.com
```

Log Monitoring

Live Monitoring of System Logs

```
tail -f /var/log/syslog  # Follow logs in real-time
journalctl -f  # Systemd logs
```

Checking Kernel Logs

```
dmesg | tail
```

Disk and Storage Management in Linux

Introduction to Disk and Storage Management

Managing disks and storage efficiently is crucial for system performance and stability. Linux provides various commands to monitor, partition, format, mount, and manage disk storage.

Index of Commands Covered

Viewing Disk Information

- `lsblk` – Display block devices
- `fdisk -l` – List disk partitions
- `blkid` – Show UUIDs of devices
- `df -h` – Check disk space usage
- `du -sh /path` – Show size of a directory

Partition Management

- `fdisk /dev/sdX` – Create and manage partitions
- `parted /dev/sdX` – Alternative to `fdisk` for GPT disks
- `mkfs.ext4 /dev/sdX1` – Format a partition as ext4
- `mkfs.xfs /dev/sdX1` – Format a partition as XFS

Mounting and Unmounting

- `mount /dev/sdX1 /mnt` – Mount a partition
- `umount /mnt` – Unmount a partition
- `mount -o remount,rw /mnt` – Remount a partition as read-write

Logical Volume Management (LVM)

- `pvcreate /dev/sdX` – Create a physical volume
- `vgcreate vg_name /dev/sdX` – Create a volume group
- `lvcreate -L 10G -n lv_name vg_name` – Create a logical volume
- `mkfs.ext4 /dev/vg_name/lv_name` – Format an LVM partition
- `mount /dev/vg_name/lv_name /mnt` – Mount an LVM partition

Swap Management

- `mkswap /dev/sdX` – Create a swap partition
- `swapon /dev/sdX` – Enable swap space
- `swapoff /dev/sdX` – Disable swap space

Viewing Disk Information

Using `lsblk`

List all block devices:

```
lsblk
```

Using `fdisk`

View partition details:

```
fdisk -l
```

Using `df`

Check available disk space:

```
df -h
```

Using `du`

Find the size of a directory:

```
du -sh /var/log
```

Partition Management

Creating a Partition with `fdisk`

```
fdisk /dev/sdX
```

Follow the interactive prompts to create a partition.

Formatting a Partition

Format as ext4:

```
mkfs.ext4 /dev/sdX1
```

Format as XFS:

```
mkfs.xfs /dev/sdX1
```

Mounting and Unmounting

Mount a Partition

```
mount /dev/sdX1 /mnt
```

Unmount a Partition

```
umount /mnt
```

Remount a Partition

```
mount -o remount,rw /mnt
```

LVM Management

Create a Physical Volume

```
pvcreate /dev/sdX
```

Create a Volume Group

```
vgcreate vg_name /dev/sdX
```

Create a Logical Volume

```
lvcreate -L 10G -n lv_name vg_name
```

Format and Mount the Logical Volume

```
mkfs.ext4 /dev/vg_name/lv_name
mount /dev/vg_name/lv_name /mnt
```

Swap Management

Create a Swap Partition

```
mkswap /dev/sdX
```

Enable Swap

```
swapon /dev/sdX
```

Disable Swap

```
swapoff /dev/sdX
```