**PROJECT P1**

**ON**

**LINUX STORAGE**

**ADMINISTRATION**

**Guided by**

**Zakir Hussian**

**Team Members:**

**Mayuri**

**Rani**

**Tejas**

**Yugandhar**

**Siddharth**

**Imran**

**Aditya**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SR.NO.** | **TITLE** | **PAGE NO.** |
| **1** | **INTRODUCTION** | **2** |
| **2** | **FILE SYSTEM AND TYPES** | **3** |
| **3** | **REAL TIME SCENARIO BASED STORAGE ADMINISTRATION** | **6** |
| **4** | **TYPES OF STORAGE** | **7** |
| **5** | **DISK PARTITIONING IN LINUX** | **10** |
| **6** | **LVM (Logical Volume Manager)** | **13** |
| **7** | **RAID** | **16** |
| **8** | **PURPOSE** | **22** |
| **9** | **KEY FEATURES AND ADVANTAGES** | **23** |
| **10** | **CONCLUSION** | **26** |

**INTRODUCTION**

The **Linux Storage Administration** project is designed to provide practical experience in managing and configuring storage systems in a Linux environment, with a focus on CentOS. Effective storage management is a fundamental aspect of system administration, as it ensures that data is stored efficiently, remains accessible, and is protected against potential failures.

In this project, we explore various aspects of storage administration, including disk partitioning, Logical Volume Management (LVM), file system creation, and disk quota management. We also delve into advanced topics such as RAID configuration and performance optimization to ensure a robust and scalable storage infrastructure.

The project begins with setting up the necessary environment and tools, followed by the implementation of storage management techniques. Key tasks include creating and managing partitions, configuring physical and logical volumes, and setting up automated mount points for persistent storage. Furthermore, we demonstrate how to monitor storage usage and implement disk quotas to prevent overutilization of resources.

This project not only showcases the practical application of storage administration commands and tools but also highlights troubleshooting strategies and best practices for maintaining a stable and efficient storage environment. By the end of this project, we aim to provide a comprehensive understanding of storage management, which is crucial for ensuring data reliability and system performance in a professional IT setting.

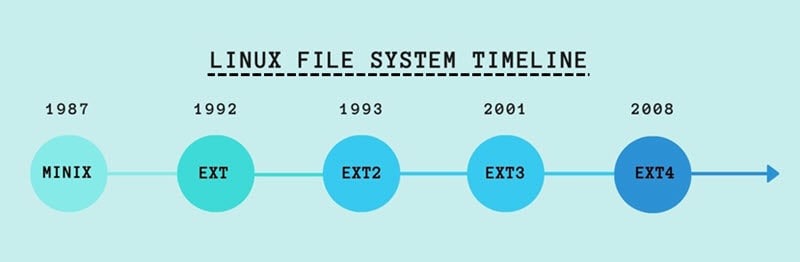
**Objective:** Briefly state the purpose of the project. For example, "The objective of this project is to demonstrate proficiency in managing and administering storage in a Linux environment using CentOS."

**Scope:** Describe what will be covered in the report, including types of storage management and administrative tasks.

**WHAT IS FILE SYSTEM AND TYPES OF FILE SYSTEM?**

A file system is a set of processes that controls how, where and when data is stored and retrieved from a storage device. An efficient file system is essential for everyday system processes.

The Linux kernel supports various file systems, but the most commonly used is the ext4 file system



FS stands for File System, which is a way of organizing and storing data on a disk. It manages how data is stored, retrieved, and updated on storage devices like hard drives, SSDs, and USB drives.

**EXT3 (Third Extended File System):**

Features:

Introduced journaling to reduce the risk of data corruption in case of a system crash.Limited scalability compared to modern file systems.

Uses: Suitable for older Linux distributions, stable for general-purpose use but less optimal for high-performance applications.

**EXT4 (Fourth Extended File System):**

Features:

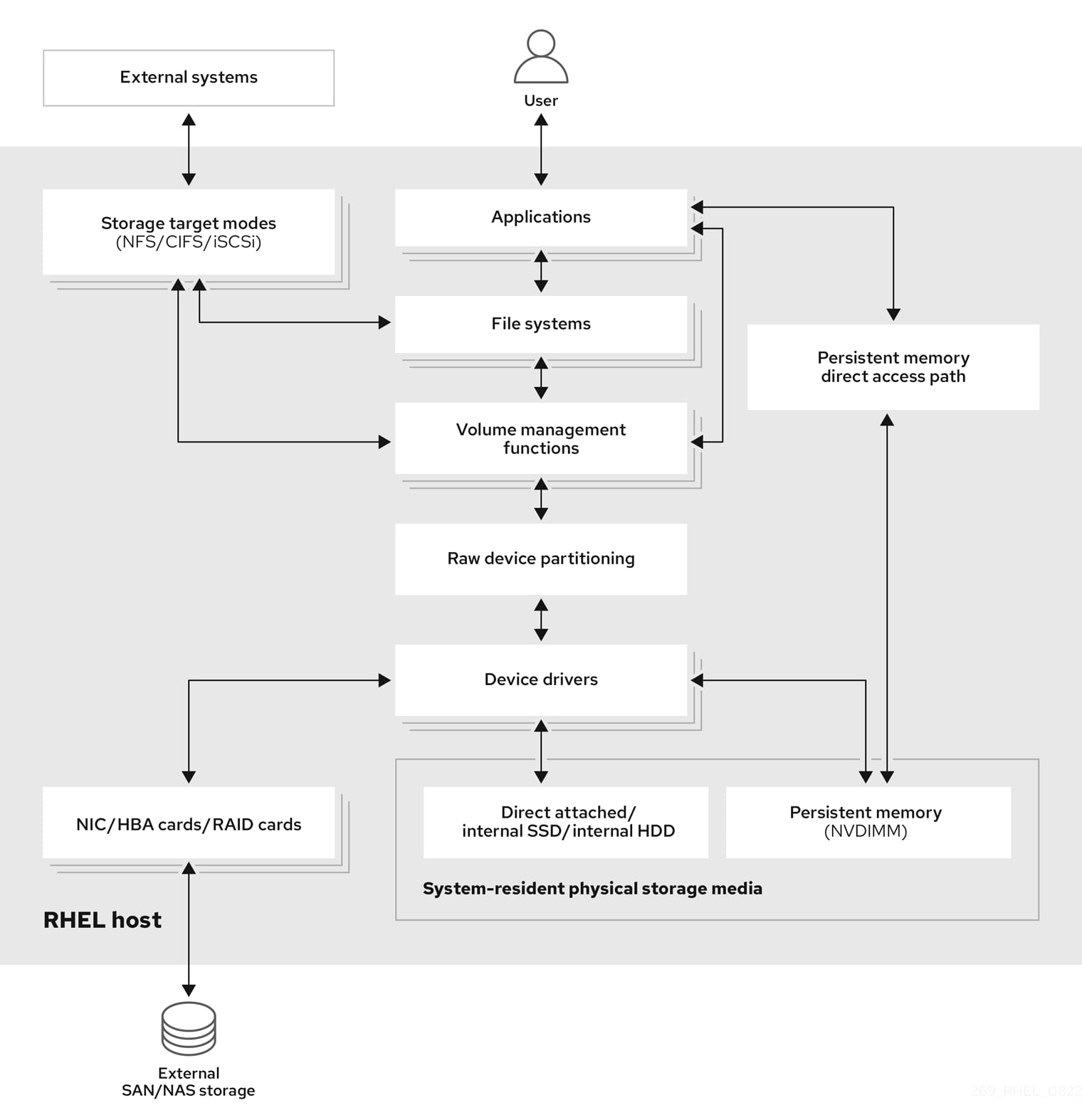
Successor to EXT3, offering improvements like larger file and volume sizes (up to 1 exabyte). Better performance with delayed allocation, which helps with more efficient disk space usage. Fast file system checks and journaling for crash recovery.

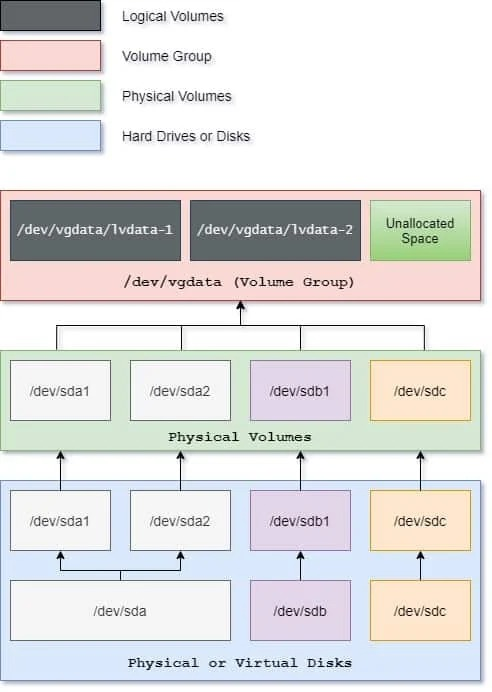
Uses: Widely used in modern Linux distributions (e.g., Ubuntu, Fedora) for general-purpose workloads and high-performance tasks.

**XFS (Extended File System):**

Features:

High-performance 64-bit journaling file system with a focus on scalability. Excellent for handling large files and managing high-performance workloads. Supports parallel input/output operations, which is useful for multi-threaded applications. Uses: Ideal for systems that handle large files and high-throughput environments like databases, multimedia processing, and scientific computing.





**REAL TIME SCENARIO BASED STORAGE ADMINISTRATION:**

In a real-world scenario, consider an e-commerce company managing massive amounts of customer data, including personal details, order histories, payment information, and product catalog. The storage administrator's role becomes critical in ensuring that all this data is stored, managed, and secured efficiently.

The company uses a hybrid cloud environment to store structured and unstructured data. A storage administrator is tasked with optimizing the use of both on-premises storage (for mission-critical applications) and cloud-based solutions (for scalability and cost-effectiveness). Daily operations involve monitoring storage usage, ensuring data availability, performing backups, and managing storage capacity.

One day, a sudden spike in customer activity during a promotional event leads to a surge in data generation. The storage administrator needs to quickly allocate more storage to handle the influx without impacting system performance. They dynamically provision additional cloud storage through the company’s cloud service provider to ensure there is no disruption.

Simultaneously, the administrator notices a slower-than-usual database performance. Upon investigation, they discover that outdated and redundant data in the on-premises storage is consuming valuable space and affecting retrieval times. They initiate a data archiving process, moving older data to cheaper, long-term storage solutions in the cloud while retaining frequently accessed data locally for faster performance.

Additionally, the administrator must ensure that all stored data complies with relevant privacy regulations (like GDPR) by configuring encryption for sensitive customer data and ensuring regular audits of storage systems for vulnerabilities.

By balancing storage needs in a dynamic, hybrid environment, the administrator ensures high performance, scalability, cost-efficiency, and security, all while maintaining business continuity and compliance with legal requirements.

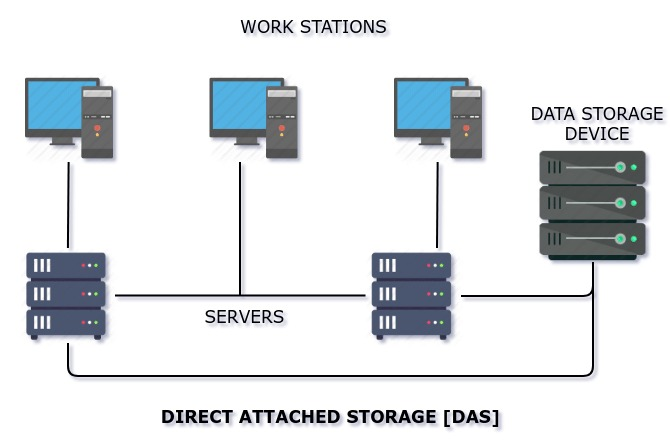
**TYPES OF STORAGE:**

**1)Direct-Attached Storage (DAS)**

DAS is directly connected to a computer or server without a network in between. Examples include internal hard drives, SSDs, and external USB drives.

Pros: High performance and low latency since the storage is directly connected. It’s also relatively simple to set up and manage.

Cons: Limited scalability and flexibility. It cannot be easily shared across multiple systems.

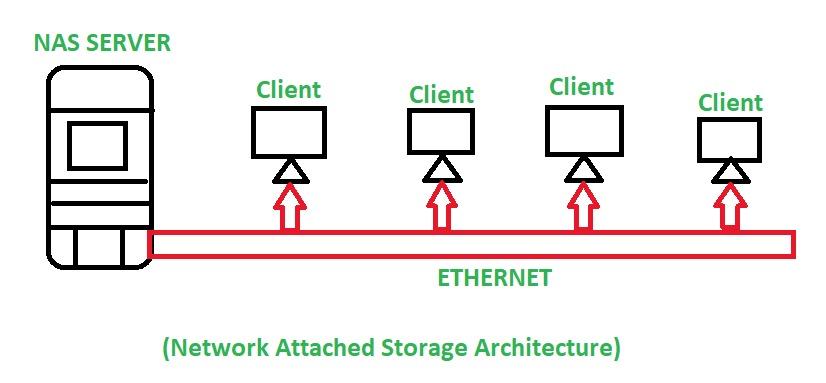


**2)Network-Attached Storage (NAS)**

Description: NAS is a dedicated file storage device that connects to a network, allowing multiple users and devices to access the data.

Pros: Easy to share files across multiple devices. It often includes features like RAID for redundancy and data protection.

Cons: Network performance can be a bottleneck. It may require more complex setup and management compared to DAS.



**3)Storage Area Network (SAN)**

SAN is a high-speed network that provides access to consolidated block-level storage. It is typically used in enterprise environments.

Pros: High performance and scalability. It supports advanced features like clustering and disaster recovery.

Cons: Expensive and complex to set up and manage. Requires specialized hardware and expertise.

**4)Cloud Storage**

Cloud storage involves storing data on remote servers accessed via the internet. Providers like AWS, Google Cloud, and Azure offer these services.

Pros: Highly scalable and accessible from anywhere. It reduces the need for on-premises hardware.

Cons: Ongoing costs can add up. Data transfer speeds depend on internet connectivity1.Each of these storage types has its own advantages and is suited to different scenarios.

For example, DAS might be ideal for a single server with high-performance needs, while NAS or SAN could be better for environments requiring shared access and high availability.

**DISK PARTITIONING IN LINUX**

Disk Partitioning is the process of dividing a disk into one or more logical areas, often known as partitions, on which the user can work separately. It is one step of disk formatting. If a partition is created, the disk will store the information about the location and size of partitions in the partition table. With the partition table, each partition can appear to the operating system as a logical disk, and users can read and write data on those disks. The main advantage of disk partitioning is that each partition can be managed separately.

**Why do we need it?**

-To upgrade Hard Disk (to incorporate a new Hard Disk into the system)

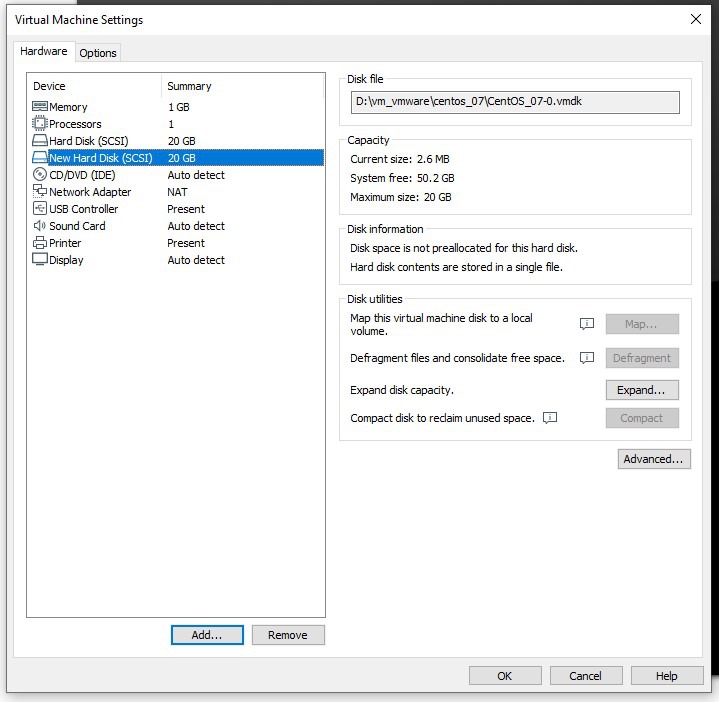
-Dual Booting (Multiple Operating Systems on the same system)

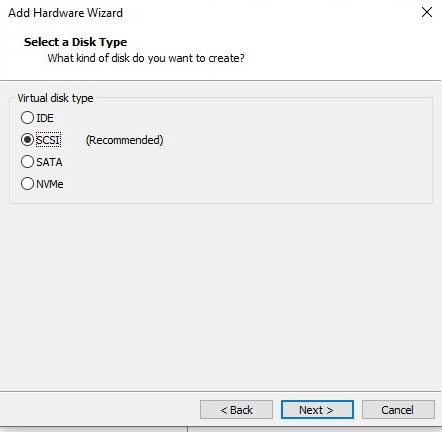
-Efficient disk management

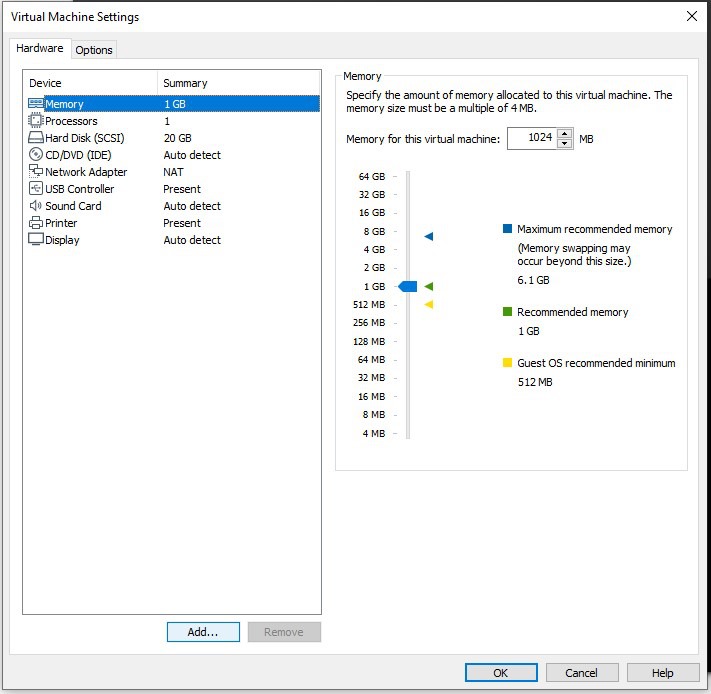
-Ensure backup and security.

-Work with different File Systems using the same system.









**LVM (Logical Volume Manager)** is a system used for managing disk storage in a more flexible and efficient way compared to traditional partitioning. It allows for dynamic resizing, creation, and management of logical volumes, which can be used in place of conventional disk partitions.

Key Concepts of LVM:

1 Physical Volumes (PVs):

These are the actual storage devices, such as hard drives or partitions, that LVM uses. You can combine multiple physical volumes into a pool of storage.

2 Volume Groups (VGs):

A volume group is created by combining one or more physical volumes. It acts as a pool of storage space from which logical volumes can be allocated.

Think of a VG as a large virtual container made up of several physical disks or partitions.

3 Logical Volumes (LVs):

Logical volumes are created within a volume group and are similar to partitions but much more flexible. You can resize, create, or remove logical volumes without worrying about the physical layout of the storage.

LVs can be easily resized on-the-fly (even while the system is running), making them ideal for systems where the storage needs fluctuate.

Common Use Cases for LVM:

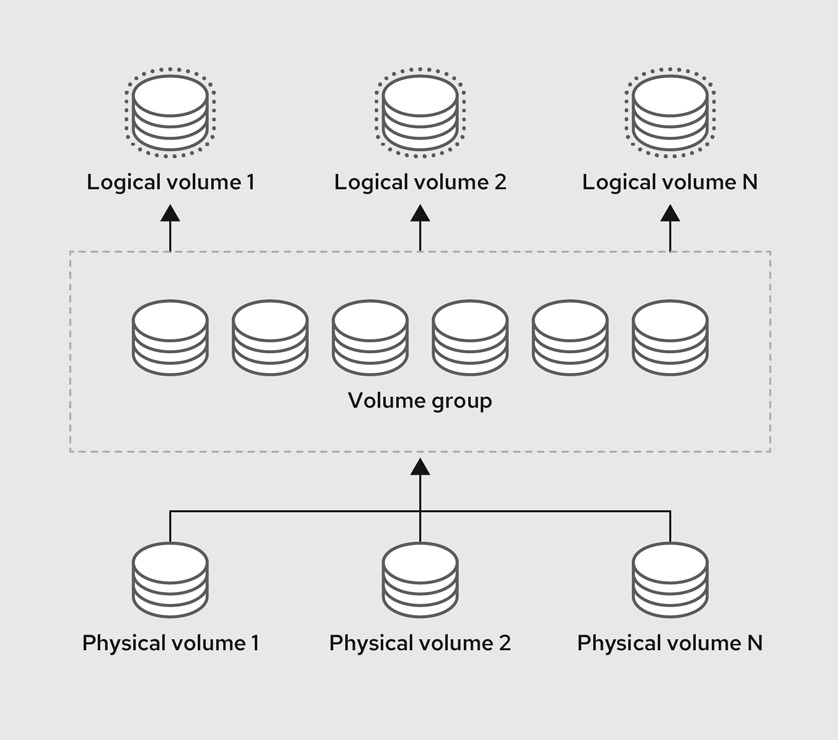
* Servers and Databases: Where storage needs to grow dynamically and manage large amounts of data efficiently.

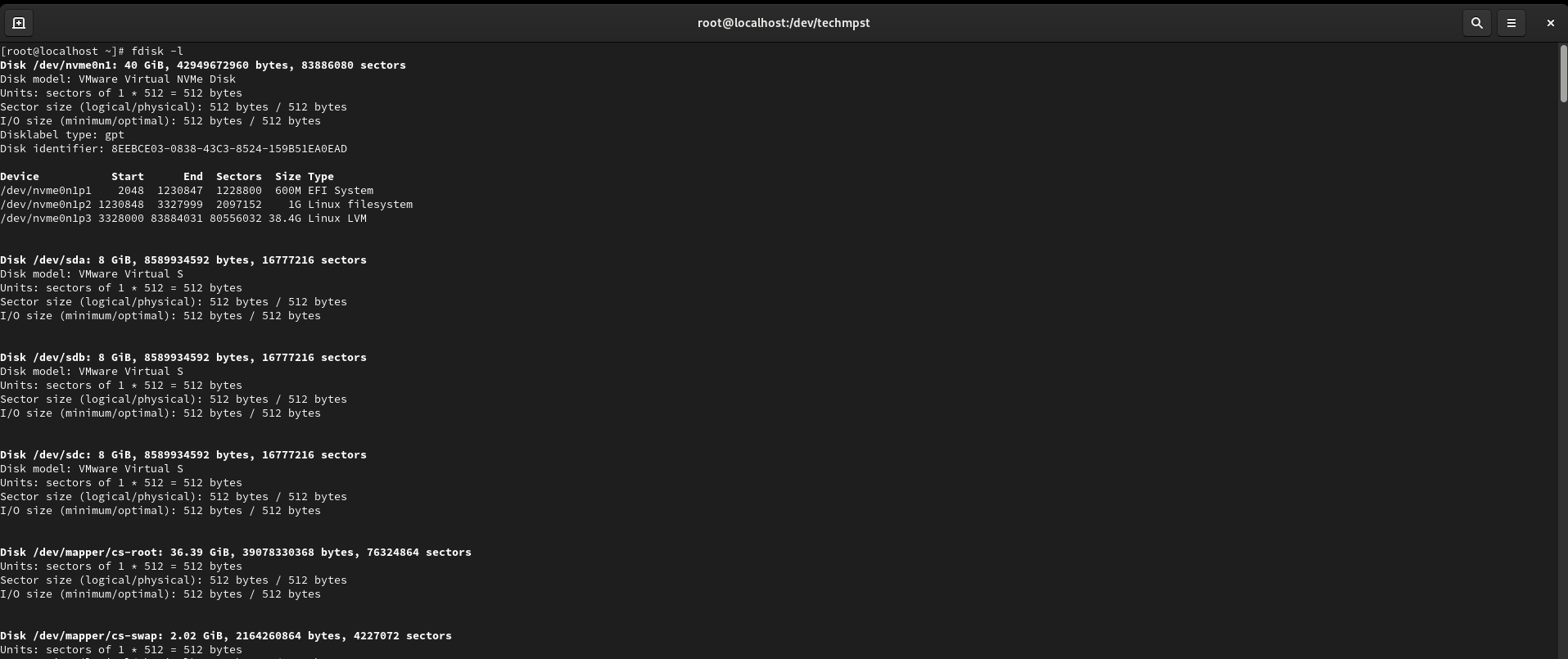
Workstations: For users who need flexible partitioning, such as developers or data scientists.

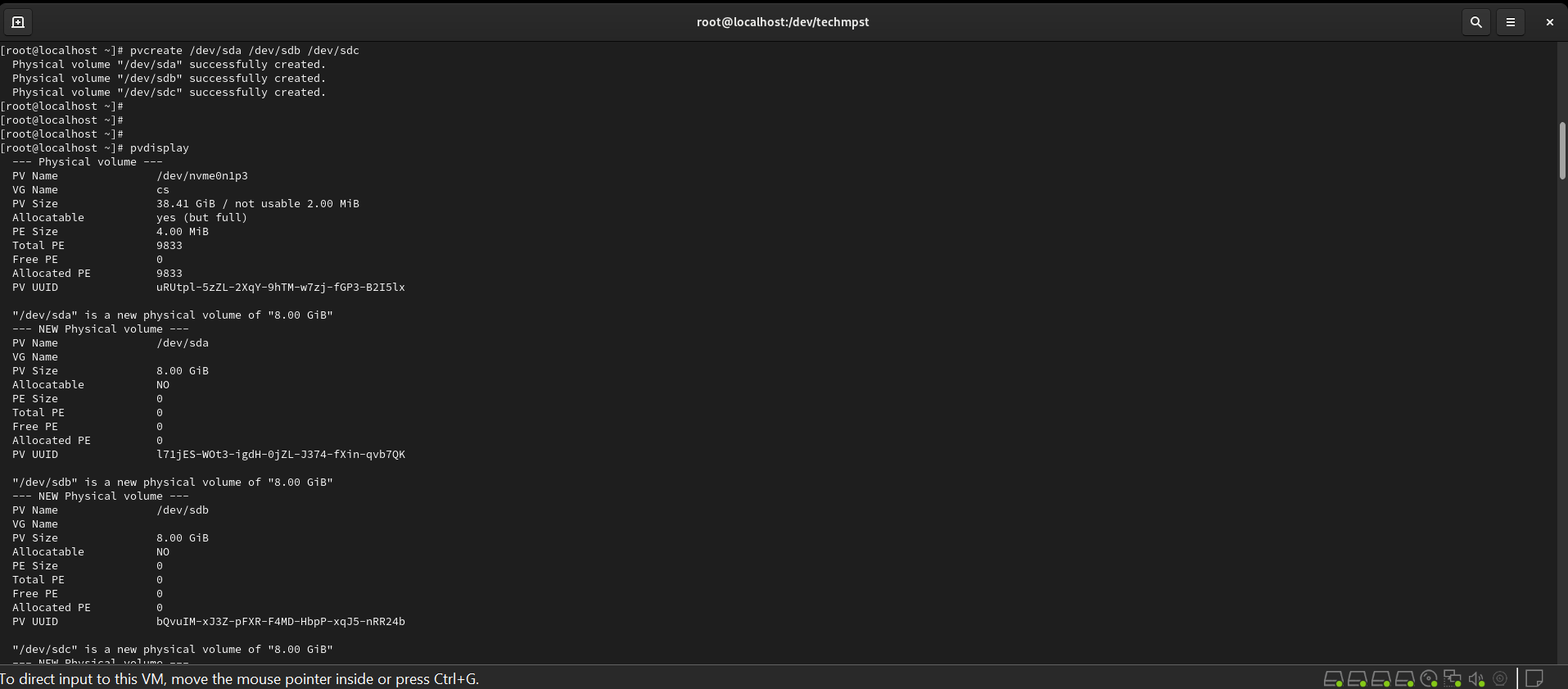
* Virtual Machines (VMs): To easily allocate and manage disk space between multiple virtual environments.

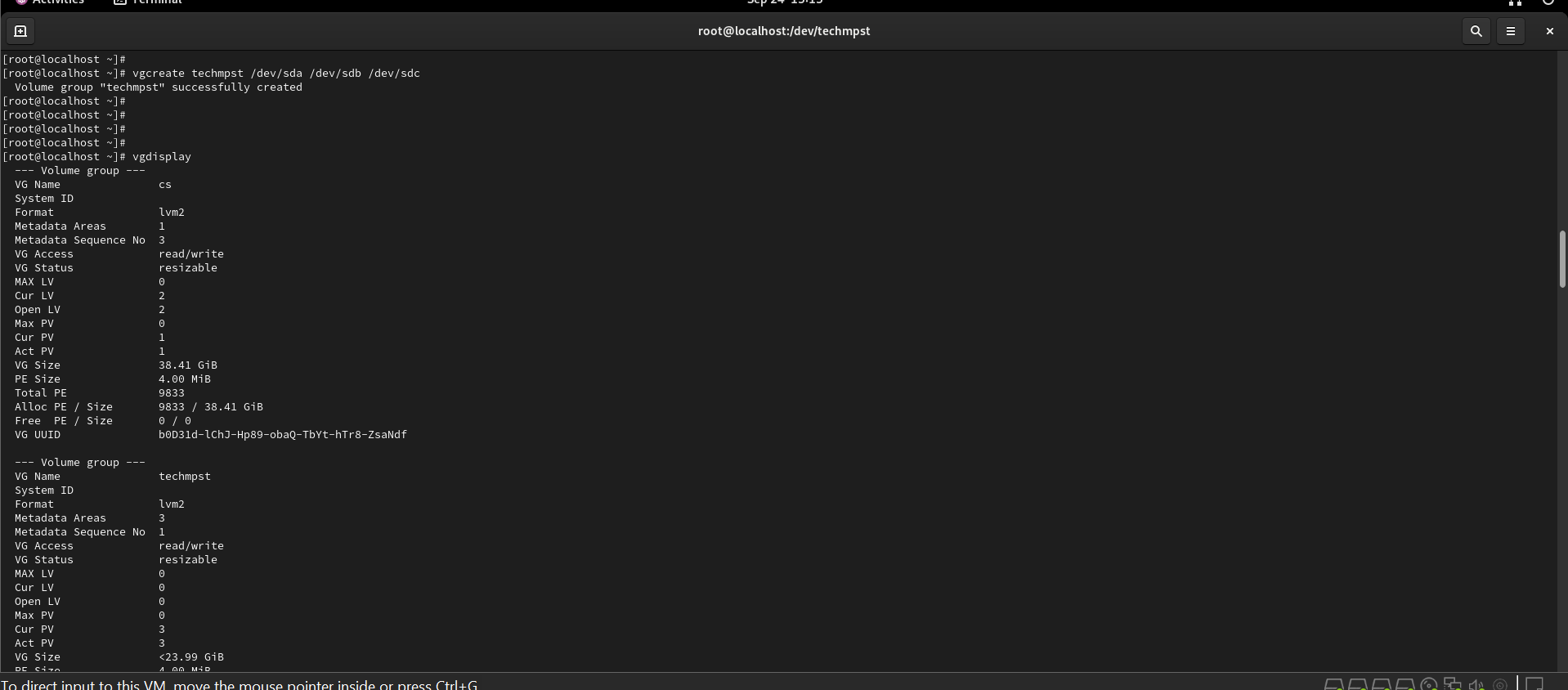
LVM in Action:

* Example 1: You can increase the size of a logical volume if a filesystem is running out of space by simply adding a new disk to the volume group and extending the logical volume.
* Example 2: You can take a snapshot of a logical volume before making a major change, such as upgrading software, ensuring you can roll back to the snapshot if necessary.









**RAID**

**RAID** contains groups or sets or Arrays. A combine of drivers make a group of disks to form a RAID Array or RAID set. It can be a minimum of 2 number of disk connected to a raid controller and make a logical volume or more drives can be in a group. Only one Raid level can be applied in a group of disks. Raid are used when we need excellent performance. According to our selected raid level, performance will differ. Saving our data by fault tolerance & high availability.

**TYPES OF RAID**:

**RAID 0 (Striping):**

\* Concept: Data is split into blocks and written across multiple disks in a striped pattern. For example, block 1 goes to disk 1, block 2 goes to disk 2, block 3 goes to disk 1, and so on. This allows for parallel read/write operations, significantly boosting performance.

\* Strengths:

\* High Performance: Offers the fastest read and write speeds among RAID levels.

\* Easy Setup: Relatively simple to configure.

\* Weaknesses:

\* No Redundancy: If one disk fails, all data is lost.

\* Risk Aversion: Not suitable for critical data or applications that require high availability.

\* Use Cases:

\* High-Performance Applications: Video editing, gaming, scientific simulations, where speed is paramount.

\* Temporary Data: Storage for temporary files, scratch space, or logs where data loss is acceptable.

**RAID 1 (Mirroring):**

\* Concept: Data is duplicated exactly onto two or more disks. Any write operation is applied to all disks in the array.

\* Strengths:

\* High Reliability: Offers excellent data protection. If one disk fails, the other takes over seamlessly.

\* Fast Read Operations: Reads can be served from either disk, improving read performance.

\* Weaknesses:

\* Slower Write Performance: Writing requires updates to multiple disks, resulting in slower write speeds.

\* Disk Space Inefficiency: Requires twice the amount of disk space as the amount of data stored.

\* Use Cases:

\* Critical Systems: Databases, servers, or systems where data loss is unacceptable. \* High-Availability Applications: Where uninterrupted operation is essential, like financial systems.

**RAID 4 (Striping with Dedicated Parity Disk):**

\* Concept: Data is striped across multiple disks, and a dedicated disk is used for parity information. The parity disk stores a parity block that allows for data recovery if a data disk fails.

\* Strengths:

\* Good Read Performance: Read operations are fast since the parity disk is not involved in reading.

\* Fault Tolerance: Can tolerate one disk failure.

\* Weaknesses:

\* Slower Write Performance: Write performance is generally slower than RAID 5 because all write operations involve the parity disk.

\* Bottleneck on the Parity Disk: The parity disk becomes a single point of failure, as it handles all parity operations.

\* Use Cases:

\* Limited use: RAID 4 is generally less common than RAID 5. It might be used in situations where write performance is less critical than read performance.

**RAID 5 (Striping with Distributed Parity):**

\* Concept: Data is striped across multiple disks, and a parity block (calculated from the data) is distributed across all disks. The parity block enables data recovery if a disk fails.

\* Strengths:

\* Good Performance: Offers a good balance of performance and reliability.

\* Fault Tolerance: Can tolerate one disk failure.

\* Efficient Disk Utilization: Uses disk space more efficiently than RAID 1, with a smaller overhead for parity.

\* Weaknesses:

\* Slower Write Performance: The parity calculation adds overhead to write operations, making them slightly slower than RAID 0.

\* Slower Reconstruction: Rebuilding the RAID array after a disk failure can take significant time.

\* Use Cases:

\* General-Purpose Servers: Web servers, file servers, and applications that require a mix of speed and reliability.

\* Data Storage: Storing large amounts of data where some redundancy is necessary.

RAID (Redundant Array of Independent Disks) offers several advantages over using individual disks:

1. Performance Enhancement:

\* Increased Read/Write Speeds: RAID levels like RAID 0 and RAID 10 significantly improve read and write speeds by allowing parallel data access across multiple disks. This is particularly beneficial for applications that require fast data transfer, such as video editing, database operations, and high-performance computing.

2. Data Protection and Fault Tolerance:

\* Data Redundancy: RAID levels like RAID 1, RAID 5, and RAID 6 provide redundancy by storing copies of data or parity information. This redundancy ensures that data can be recovered even if one or more disks fail.

\* Improved Reliability: RAID arrays increase the reliability of storage systems by mitigating the risk of data loss due to single disk failures. This is essential for critical systems where downtime or data loss is unacceptable.

3. Improved Disk Utilization:

\* Efficient Disk Space: RAID levels like RAID 5 and RAID 6 use disk space more efficiently than RAID 1, reducing the amount of storage space required for data redundancy.

4. Scalability:

\* Flexible Capacity: RAID allows you to combine multiple disks to create larger storage arrays, making it easy to scale storage capacity as your needs grow.

5. Cost-Effectiveness:

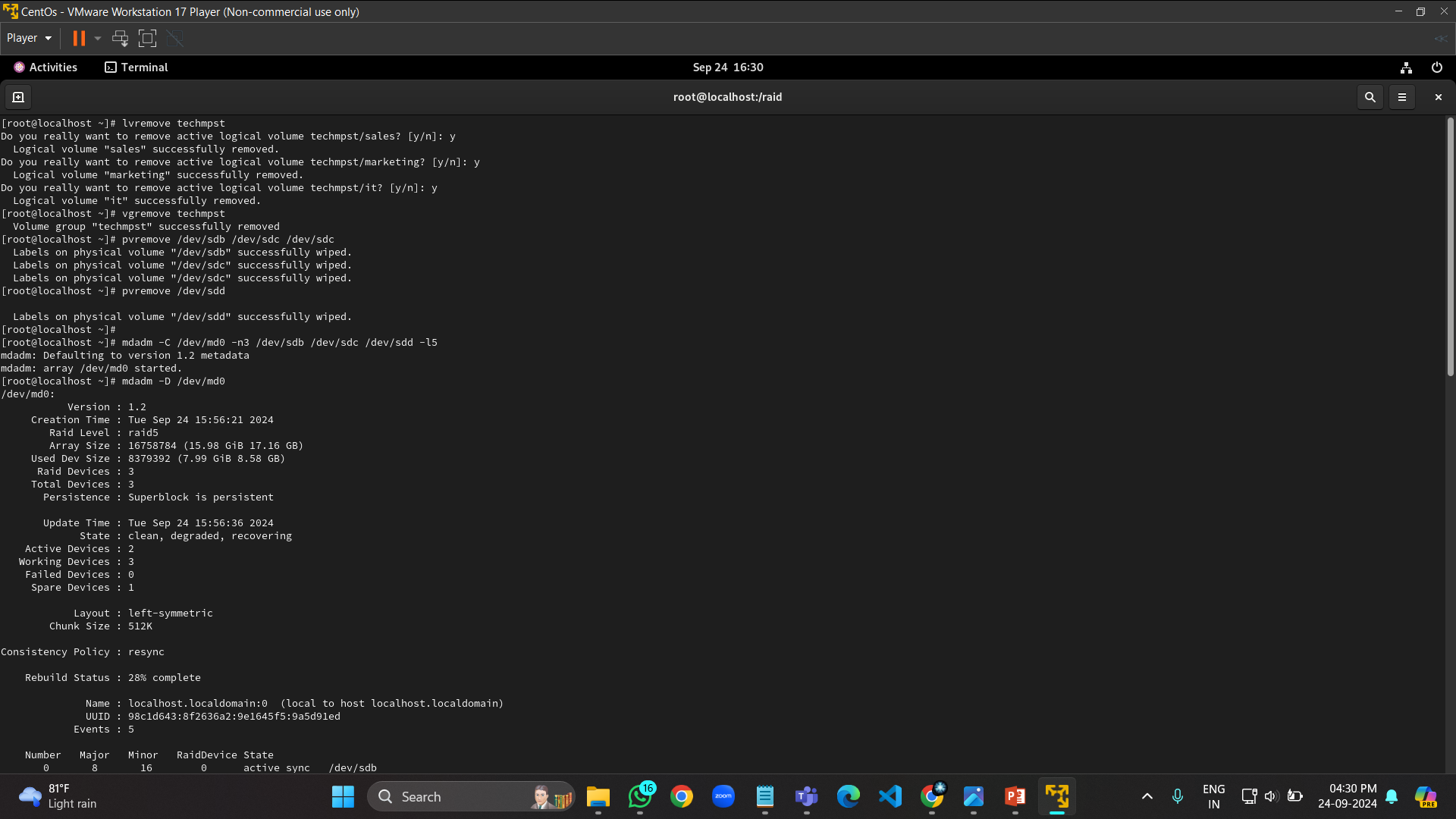
\* Lower Overall Cost: RAID can be more cost-effective than using a single large disk for storing data. This is especially true for high-capacity storage requirements, as multiple smaller disks can be combined to achieve the desired capacity at a lower cost.

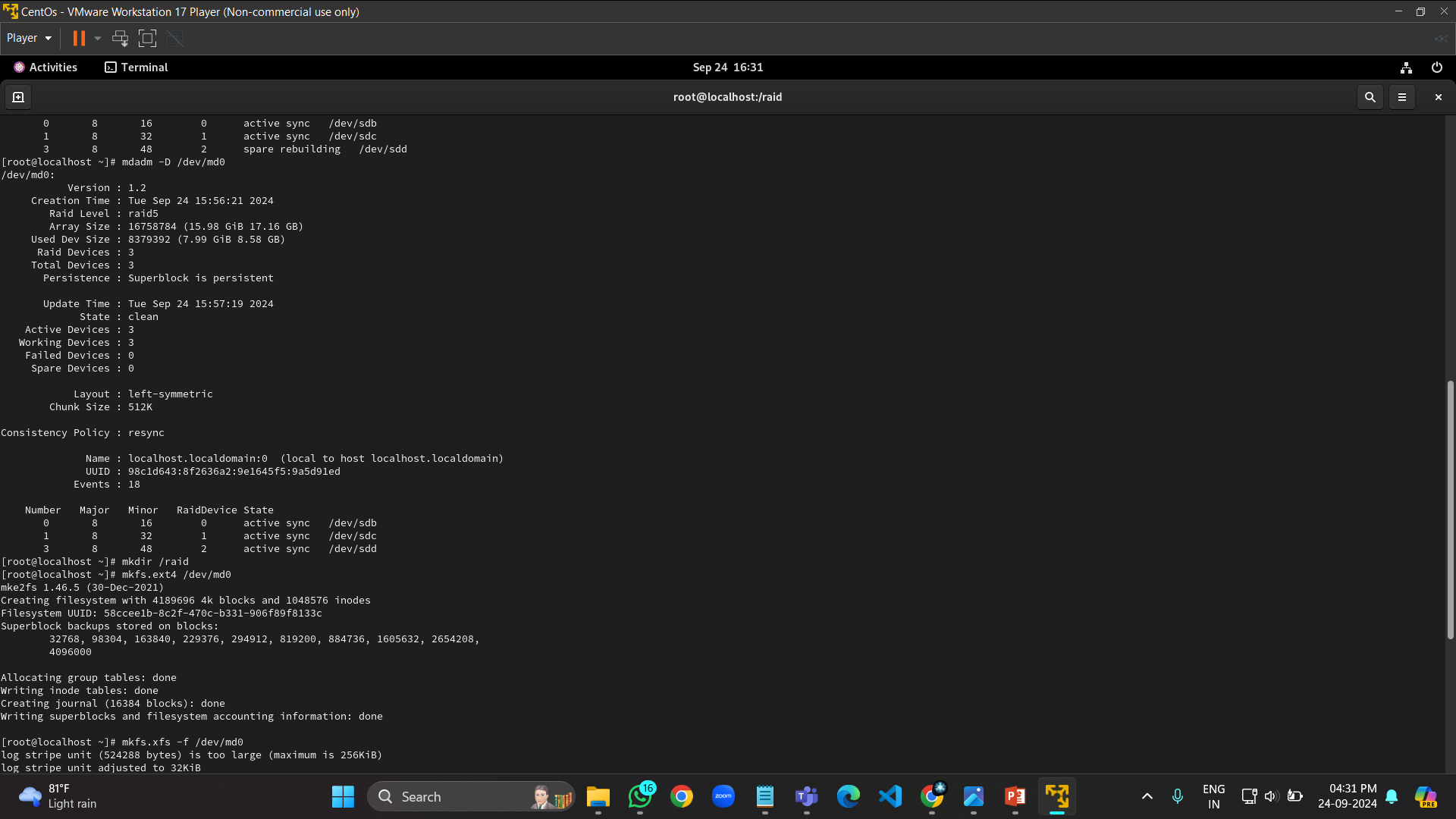
6. Enhanced Data Availability:

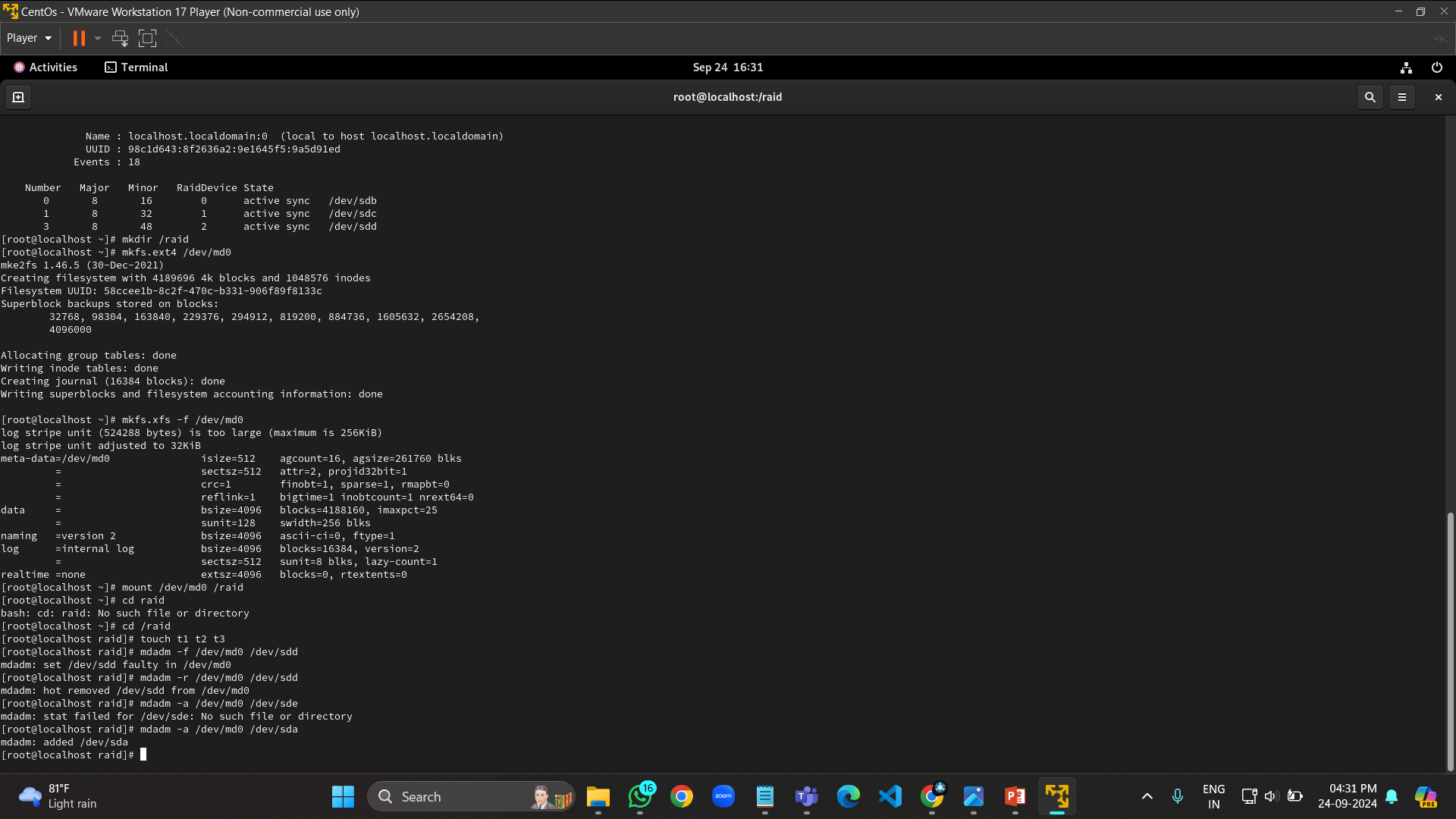
\* High Availability: RAID levels like RAID 1 and RAID 10 offer high availability, ensuring that data remains accessible even if a disk fails. This is crucial for applications that require continuous operation, such as databases and web servers.

7. Simplified Management:

\* Single Logical Unit: RAID arrays are presented as a single logical unit to the operating system, simplifying storage management and making it easier to access and manage data.







**CABLES USE IN STORAGE ADMINISTRATION:**

Fiber Optic: High bandwidth, used in SANs for long-distance connections.

Twisted Pair Cable: Commonly used in Ethernet networks, provides reliable, cost-effective data transmission over short distances, often for NAS storage setups.

**PURPOSE**

1.Load Balancing :

Load balancing in Linux storage administration refers to the distribution of I/O (input/output) operations across multiple storage devices or paths to ensure optimal utilization, redundancy, performance, and reliability. This is crucial in environments where there are multiple paths between the system and the storage devices, such as in SAN (Storage Area Networks) configurations. Without load balancing, certain paths could be over-utilized while others remain under-utilized, leading to bottlenecks and reduced performance.

1. Multipath I/O (MPIO):

Multipath I/O (MPIO) is a technique that allows a Linux system to use multiple physical paths to access storage devices. It provides two main advantages:

Redundancy: In case one path fails, others can take over to prevent downtime.

Performance: Load balancing distributes I/O requests across multiple paths, optimizing bandwidth and reducing latency.

2. Components of Multipath I/O:

Multipath Devices: These are logical devices that aggregate multiple physical paths to a storage device into a single device seen by the operating system.

Path Groups: Paths are grouped based on their characteristics, and MPIO decides how to route I/O traffic within these groups.

Path Checker: This component monitors the health of each path, ensuring that only functional paths are used.

3. Path Selection and Load Balancing Policies:

The method used to distribute I/O across multiple paths is defined by the load balancing policy. Common policies include:Round-Robin: I/O is distributed evenly across all paths in a cyclical manner. This is simple and effective when paths have similar performance.

Least Queue Depth: I/O is sent to the path with the fewest active requests, ensuring no single path becomes overloaded.Failover (Active/Passive): All I/O goes through a primary path unless it fails, at which point a backup path is used.

4. Redundancy :

Redundancy in Linux storage administration can be achieved at various levels:

RAID for disk-level redundancy.Multipath I/O (MPIO) for path-level redundancy.

LVM for logical volume redundancy with mirroring or RAID.Clustered File Systems (e.g., GlusterFS, Ceph) for node-level redundancy in distributed environments.

**KEY FEATURES AND ADVANTAGES:**

Linux storage administration provides powerful and flexible tools for managing storage in enterprise environments. Here are its essential features and advantages:

1. Support for Multiple File Systems

Linux supports a wide variety of file systems, offering flexibility for different use cases:

Ext4: Default file system, widely used for its reliability, journaling, and large file support.

XFS: Ideal for high-performance and large-scale environments, with fast recovery times.

Btrfs: Offers advanced features like snapshots, self-healing, and checksums.

ZFS: Known for its robustness, scalability, data integrity features, and built-in RAID functionality.

Advantages:Freedom to choose file systems tailored to performance, data integrity, or specific use-case requirements.

2. Logical Volume Management (LVM)

LVM allows administrators to manage disk space more dynamically:

Logical Volumes (LV) can be resized (expanded or shrunk) without interrupting system operations.Volume Groups (VG): Combine multiple physical disks into a single pool of storage, simplifying management.Snapshots: Create point-in-time copies of data for backup or testing without stopping services.Advantages:Flexibility to resize storage volumes as needed.

Efficient disk space management across multiple disks.

3. RAID (Redundant Array of Independent Disks) Support

Linux natively supports software RAID, enabling administrators to configure various RAID levels:

RAID 0: Striping for performance improvement.

RAID 1: Mirroring for redundancy and fault tolerance.

RAID 5, 6: Striping with parity for a balance of performance and redundancy.

RAID 10: Combines striping and mirroring for both speed and data protection.

Advantages:

High availability and redundancy for critical storage environments.

Improved performance by distributing data across multiple disks.

4. Multipath I/O (MPIO) for Load Balancing

Multipath I/O allows the use of multiple physical paths between the system and storage devices, offering:

Redundancy: In case of a path failure, others take over automatically.

Load Balancing: Distributes I/O operations across paths to improve performance.

Advantages:

Fault tolerance and high availability.

Improved performance by balancing I/O traffic across multiple storage paths.

5. Networked Storage Support (SAN/NAS/iSCSI)

Linux supports several network-based storage solutions:

SAN (Storage Area Network): Provides block-level storage accessible over the network.

NAS (Network-Attached Storage): Offers file-based storage shared over the network.

iSCSI: Block-level storage over IP networks, enabling disk access across a network.

Advantages:

Scalability: Easy to add storage without physically modifying the system.

Centralized management of storage, simplifying backups and access control.

6. Automated Storage Management with udev

Linux uses udev to dynamically manage device files for hardware components:

Automatically detects and configures new storage devices.

Applies custom rules to handle specific storage devices based on their attributes.

Advantages:Simplifies device management with automatic configuration.

Customizable device handling for specific hardware.

7. Advanced Storage Features: Quotas and Encryption

Linux supports advanced features to secure and manage storage effectively:

Disk Quotas: Enforce limits on how much disk space users or groups can use.

Encryption: Tools like LUKS provide full disk encryption for data security.

Advantages:

Enforce storage usage policies and prevent individual users from monopolizing disk space.Protect sensitive data at rest using encryption.

8. Backup and Recovery Tools

Linux offers a variety of tools for backup and recovery:

rsync: A powerful tool for synchronizing files and directories between locations.

tar: Widely used for creating archives and backups.

dd: Low-level disk copying tool for creating exact backups or disk images.

Snapshots: With LVM or Btrfs, snapshots offer near-instant backups of data.

Advantages:Flexibility in backup strategies, from simple file copies to advanced snapshot technologies.Robust disaster recovery options through replication and backups.

9. Scalability and High Availability

Linux is ideal for scalable and high-availability storage setups:

Clustered File Systems: Support for distributed file systems like GlusterFS and Ceph for managing storage across multiple nodes.

High-Availability Clusters: Tools like Pacemaker and Corosync ensure storage availability in case of hardware failures.

Advantages:Scale storage horizontally by adding more nodes.Ensure uptime and data availability in mission-critical environments.

**RECOVERY AND BACKUP:**

**1. Full Backup**

* Definition: A complete copy of all data and files at a specific point in time.
* Pros: Simplifies the restore process as all data is available in a single backup.
* Cons: Time-consuming and requires a lot of storage space.
* Use Case: Suitable for situations where storage space is not an issue and complete backups are needed, such as before major system changes.

**2. Incremental Backup**

* Definition: Backs up only the data that has changed since the last backup (either full or incremental).
* Pros: Saves storage space and time compared to full backups.
* Cons: Restoration requires all incremental backups since the last full backup, which can be time-consuming and complex.
* Use Case: Suitable for systems with frequent changes and limited storage capacity.

**3. Differential Backup**

* Definition: Backs up only the data that has changed since the last full backup.
* Pros: Faster than full backups and more comprehensive than incremental backups for restoration.
* Cons: As the differential backups grow, they can become large and take up more space.
* Use Case: Suitable for situations where a balance between storage efficiency and restore time is required**.**

**CONCLUSION:**

The **Linux Storage Administration** project has provided valuable insights into the critical role of managing and configuring storage systems in a Linux environment. Throughout the project, we explored essential storage management techniques, including disk partitioning, Logical Volume Management (LVM), file system creation, and disk quota configuration. Each of these tasks demonstrated the practical application of administrative commands and best practices necessary for maintaining a reliable and efficient storage infrastructure.

One of the key takeaways from this project is the versatility and power of LVM in managing disk space dynamically. By creating and managing physical volumes, volume groups, and logical volumes, we were able to efficiently allocate and resize storage as needed, without impacting system availability. Additionally, configuring file systems and ensuring persistent mounts through the /etc/fstab file further solidified our understanding of file system management.

We also addressed the importance of disk quotas in a multi-user environment, where monitoring and controlling disk usage can prevent resource exhaustion and ensure fair usage. This project emphasized the need for proactive monitoring and the implementation of policies that safeguard against storage mismanagement.

Throughout the project, we encountered several challenges, such as handling disk failures and optimizing RAID configurations. These challenges provided an opportunity to apply troubleshooting skills and understand the resilience and redundancy offered by different RAID levels.

Overall, this project has significantly enhanced our understanding of Linux storage management and its impact on system performance and reliability. The skills and knowledge gained from this project are not only applicable to a wide range of IT environments but are also crucial for any aspiring system administrator. Moving forward, continuous learning and hands-on practice will be essential to mastering the intricacies of Linux storage administration and adapting to evolving technology.