### **Summary of Commands for LMV**

Here’s a summary of the key commands used throughout the process:

1. sudo pvcreate /dev/sdb — Create a physical volume.
2. sudo vgcreate my\_volume\_group /dev/sdb — Create a volume group.
3. sudo lvcreate -n my\_logical\_volume -L 10G my\_volume\_group — Create a logical volume.
4. sudo mkfs.ext4 /dev/my\_volume\_group/my\_logical\_volume — Create a filesystem.
5. sudo mount /dev/my\_volume\_group/my\_logical\_volume /mnt/my\_lvm — Mount the logical volume.
6. sudo nano /etc/fstab — Edit fstab for automatic mount.

More Detailed LVM with EXTENDED

1. **apt update**:  
   * Updates the package list to get information about the latest versions of packages available from the repositories.
2. **apt install lvm2**:  
   * Installs the LVM (Logical Volume Manager) tools (lvm2 package) to allow you to manage disk volumes.
3. **lsblk**:  
   * Lists the block devices on your system, showing all hard drives, partitions, and storage devices.
4. **pvcreate /dev/sdb**:  
   * Creates a physical volume (PV) on the /dev/sdb disk. This disk will now be part of your LVM setup.
5. **pvdisplay**:  
   * Displays detailed information about the physical volumes on your system.
6. **pvcreate /dev/sdc**:  
   * Creates another physical volume on the /dev/sdc disk, preparing it for use in LVM.
7. **vgcreate suraj\_volume\_group /dev/sdb /dev/sdc**:  
   * Creates a volume group (VG) named suraj\_volume\_group using /dev/sdb and /dev/sdc as physical volumes.
8. **vgdisplay**:  
   * Displays information about the volume group(s) on your system.
9. **lvcreate** (not listed explicitly, but inferred):  
   * A command used to create a logical volume (LV). It’s typically needed to create a logical volume within your volume group, but it's not in the history you’ve provided.
10. **lvdisplay**:  
    * Displays details about the logical volume(s) within the specified volume group.
11. **mkfs.ext4 /dev/suraj\_volume\_group/suraj\_logical\_volume**:  
    * Formats the logical volume suraj\_logical\_volume in the ext4 filesystem.
12. **blkid /dev/suraj\_volume\_group/suraj\_logical\_volume**:  
    * Shows detailed information about the logical volume, including its UUID and filesystem type.
13. **mkdir /mnt/suraj\_lvm**:  
    * Creates a directory where the logical volume will be mounted.
14. **mount /dev/suraj\_volume\_group/suraj\_logical\_volume /mnt/suraj\_lvm**:  
    * Mounts the logical volume to the directory /mnt/suraj\_lvm.
15. **df -h**:  
    * Displays disk space usage in a human-readable format, showing how much space is used and available on mounted filesystems.
16. **mount | grep /mnt/suraj\_lvm**:  
    * Checks if the logical volume is mounted correctly by filtering the mount command output for /mnt/suraj\_lvm.
17. **nano /etc/fstab**:  
    * Opens the /etc/fstab file for editing, where you can add a persistent entry for your logical volume to mount it automatically at boot.
18. **lvextend -L 20G /dev/suraj\_volume\_group/suraj\_logical\_volume**:  
    * Extends the logical volume suraj\_logical\_volume to 20GB. (You mentioned in your history that it was later reduced to 15GB.)
19. **lvextend -L 15G /dev/suraj\_volume\_group/suraj\_logical\_volume**:  
    * Reduces the previous extension and adjusts the logical volume size to 15GB.
20. **resize2fs /dev/suraj\_volume\_group/suraj\_logical\_volume**:  
    * Resizes the filesystem on the logical volume to match the new size (15GB).
21. **lvdisplay /dev/suraj\_volume\_group/suraj\_logical\_volume**:  
    * Displays the updated information of the logical volume after it was resized.
22. **df -h**:  
    * Checks again to see the updated disk space usage after resizing.

````````````````````````````````````````RAID 0````````````````````````````````````````````

Here’s a detailed guide to setting up **RAID 0 (Striping)** on a Linux system. RAID 0 combines multiple disks to form a single, faster storage unit. This guide also includes verification steps for each stage of the process.

### **Creating a RAID 0 Setup on Linux**

RAID 0 provides increased speed by splitting data into smaller parts and writing them to multiple drives simultaneously. However, it does **not** offer redundancy, so if one disk fails, all data will be lost. Below are the steps to create a RAID 0 array on Linux.

### **Pre-requisites:**

* A Linux system (Ubuntu, CentOS, etc.)
* At least two unformatted disks (e.g., /dev/sdb, /dev/sdc).
* A root or sudo user to execute commands.

### **Steps to Create a RAID 0 Array:**

#### **1. Install mdadm (RAID Management Tool):**

mdadm is the utility used to manage RAID arrays on Linux. Install it using the following commands:

**On Ubuntu/Debian:**

sudo apt update

sudo apt install mdadm

**On CentOS/RHEL:**

sudo yum install mdadm

#### **2. Prepare the Disks:**

Make sure the disks are unformatted. You can list available disks with:

sudo lsblk

Identify the disks you want to use for the RAID array (e.g., /dev/sdb and /dev/sdc).

#### **3. Create the RAID 0 Array:**

You can now create the RAID 0 array using mdadm. For example, to create a RAID 0 array named my\_raid0 with two disks /dev/sdb and /dev/sdc, use the following command:

sudo mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb /dev/sdc

* --create /dev/md0: Creates the RAID array and names it /dev/md0.
* --level=0: Specifies RAID 0.
* --raid-devices=2: Defines the number of devices (disks) in the RAID array.

**Verify RAID Array Creation:**

sudo mdadm --detail /dev/md0

This command provides details about the RAID array, including its status, number of disks, and size.

#### **4. Check RAID Array Status:**

You can also monitor the status of the RAID array creation by checking /proc/mdstat:

cat /proc/mdstat

This shows the current status of all RAID arrays and whether they are synchronizing.

#### **5. Create a Filesystem on the RAID Array:**

After creating the RAID array, you need to format it with a filesystem. For example, to create an ext4 filesystem on the RAID array /dev/md0, run:

sudo mkfs.ext4 /dev/md0

**Verify Filesystem Creation:**

sudo blkid /dev/md0

This command shows the filesystem type of the RAID array (e.g., ext4).

#### **6. Mount the RAID Array:**

To use the RAID array, you need to mount it. First, create a mount point:

sudo mkdir /mnt/my\_raid0

Now, mount the RAID array:

sudo mount /dev/md0 /mnt/my\_raid0

**Verify Mount:**

df -h

This will display disk usage for mounted filesystems, confirming that the RAID 0 array is successfully mounted.

Alternatively, use:

mount | grep /mnt/my\_raid0

This checks if the RAID array is mounted at the desired mount point.

#### **7. Make the Mount Persistent:**

To automatically mount the RAID array after a reboot, add an entry to the /etc/fstab file:

sudo nano /etc/fstab

Add the following line:

/dev/md0 /mnt/my\_raid0 ext4 defaults 0 0

**Verify fstab Entry:**

cat /etc/fstab

Ensure the RAID array is listed in /etc/fstab for automatic mounting.

#### **8. (Optional) Monitor the RAID Array:**

You can check the health and status of your RAID array periodically with:

sudo mdadm --detail /dev/md0

This will provide detailed information about the RAID array, such as its current status, devices, and RAID level.

### **Summary of Commands:**

**Install mdadm:** sudo apt install mdadm # On Ubuntu/Debian

sudo yum install mdadm # On CentOS/RHEL

**Create RAID 0 Array:** sudo mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb /dev/sdc

**Verify RAID Array:** sudo mdadm --detail /dev/md0

**Create Filesystem on RAID Array:** sudo mkfs.ext4 /dev/md0

**Mount RAID Array:** sudo mount /dev/md0 /mnt/my\_raid0

**Make Mount Persistent:** sudo nano /etc/fstab # Add /dev/md0 /mnt/my\_raid0 ext4 defaults 0 0

### **Example Workflow with Verification:**

**Create RAID 0 Array:** sudo mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb /dev/sdc

sudo mdadm --detail /dev/md0 # Verify RAID creation

**Create Filesystem:** sudo mkfs.ext4 /dev/md0

sudo blkid /dev/md0 # Verify filesystem type

**Mount RAID Array:** sudo mount /dev/md0 /mnt/my\_raid0

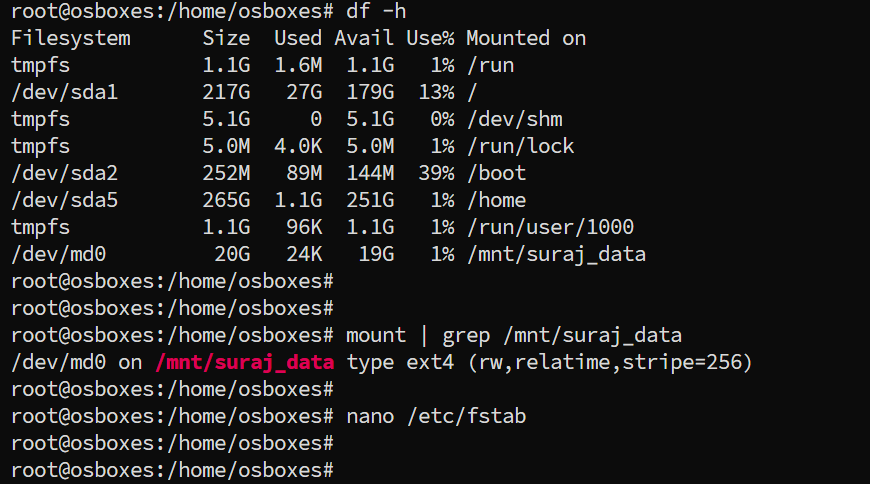
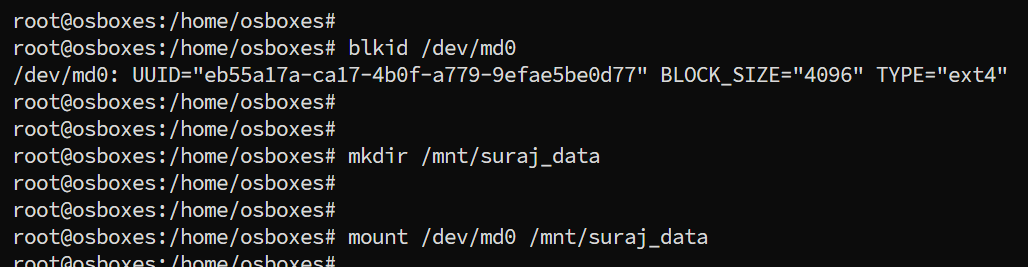
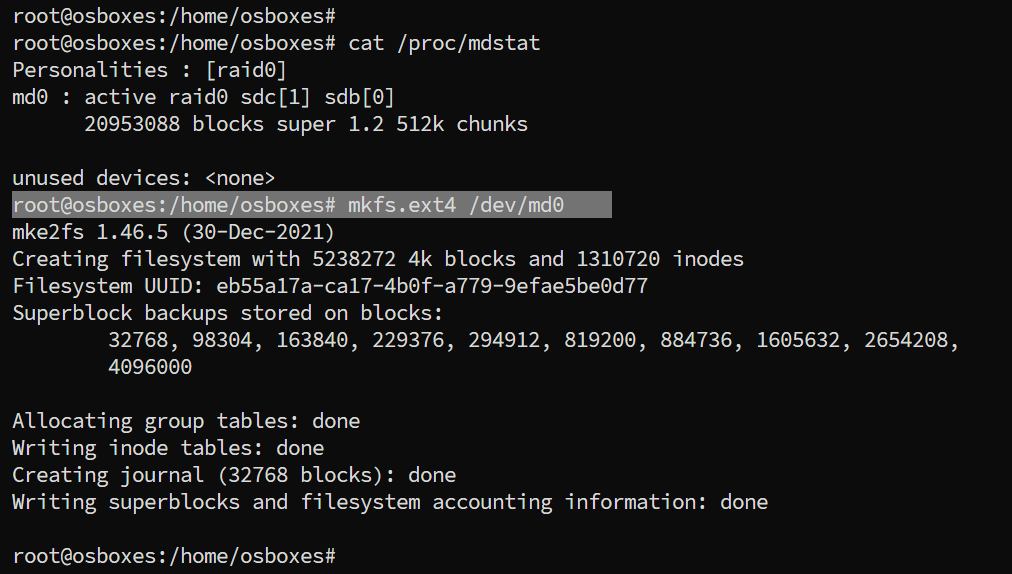
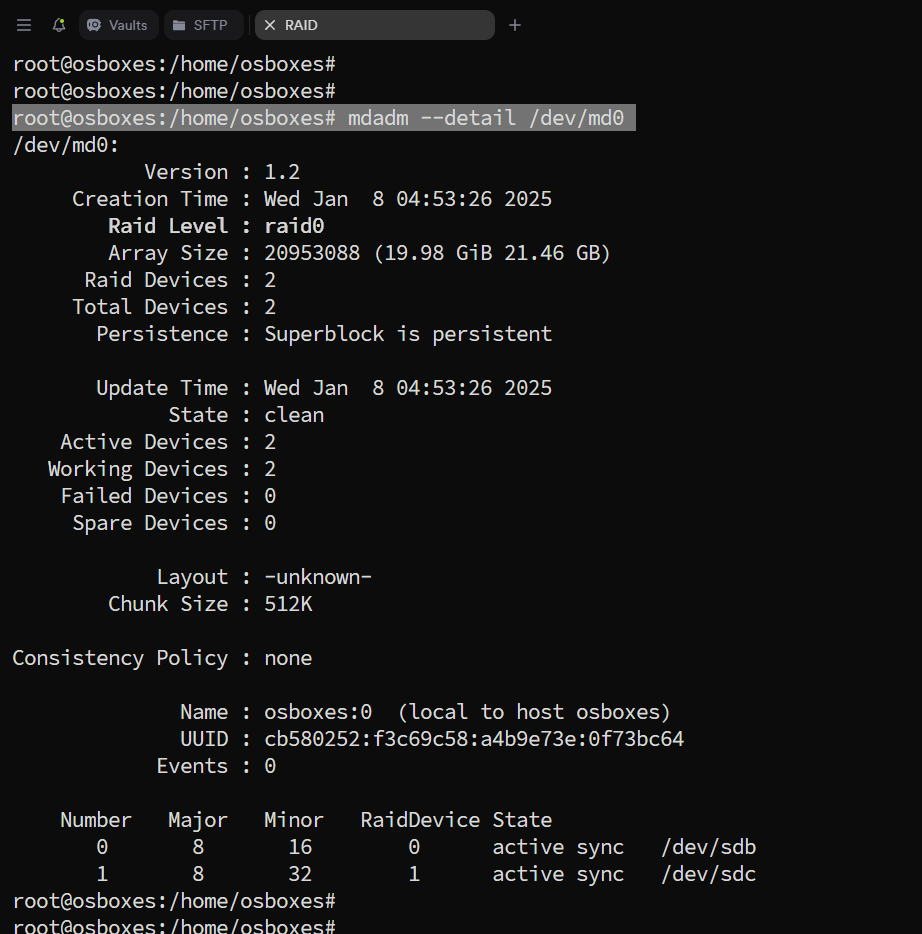
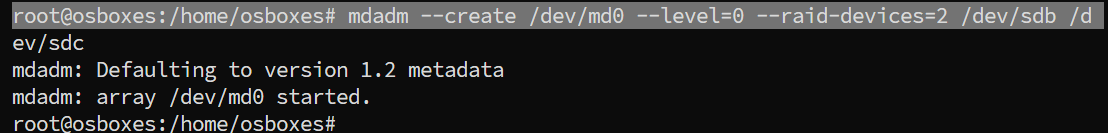
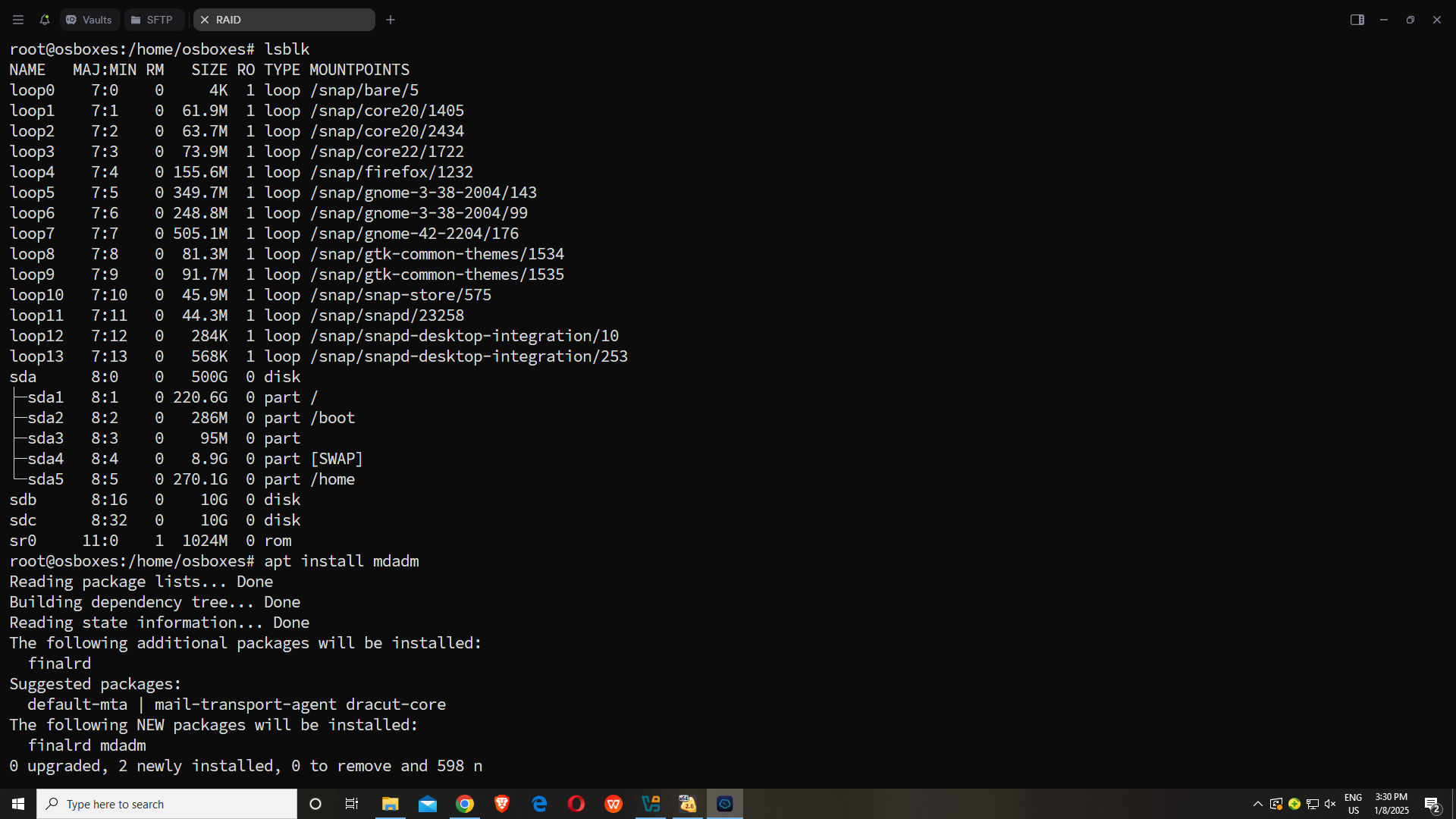
sudo df -h # Verify mount

**Make Mount Persistent (Verify fstab):** sudo nano /etc/fstab # Verify fstab entry for persistent mount

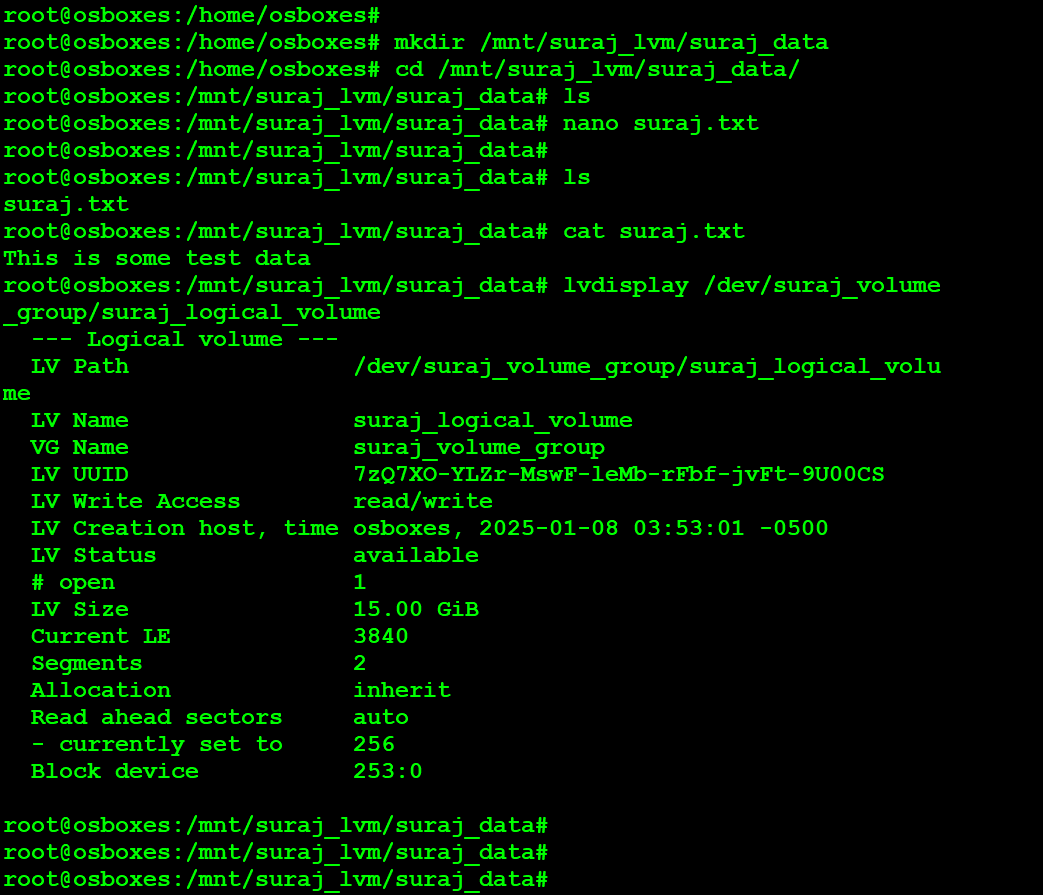
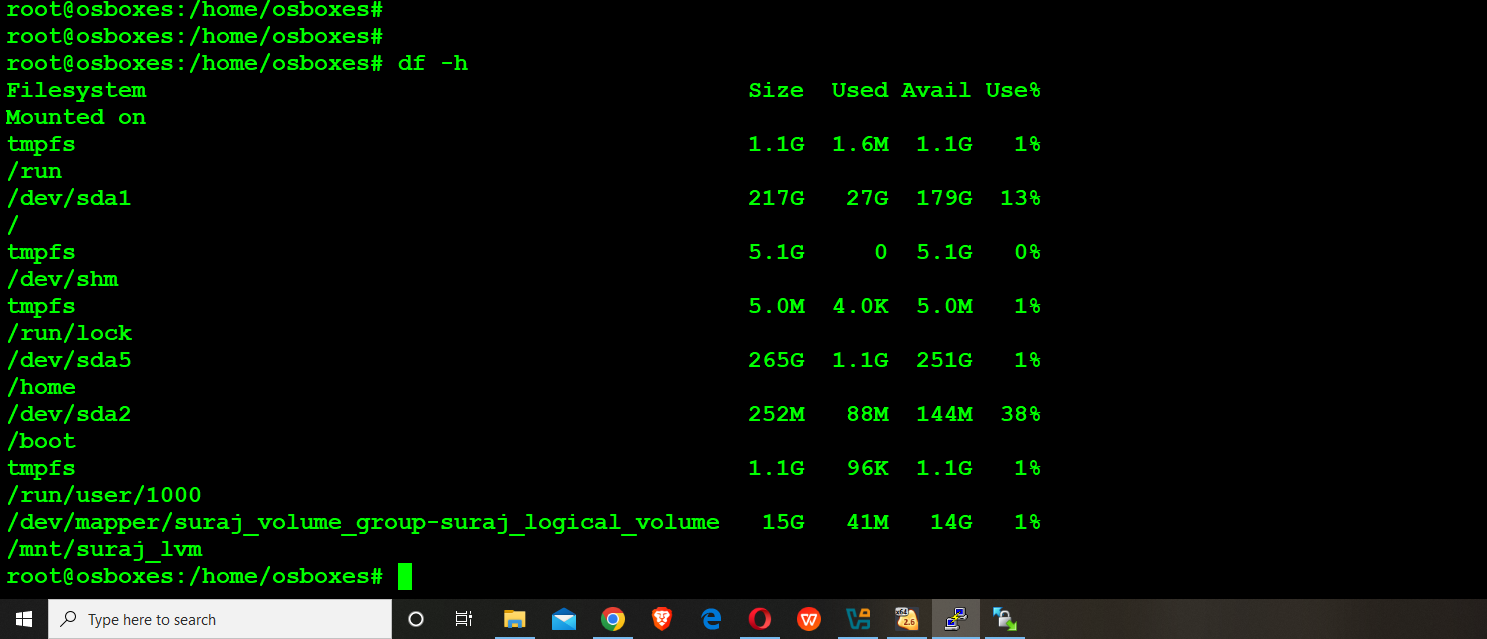
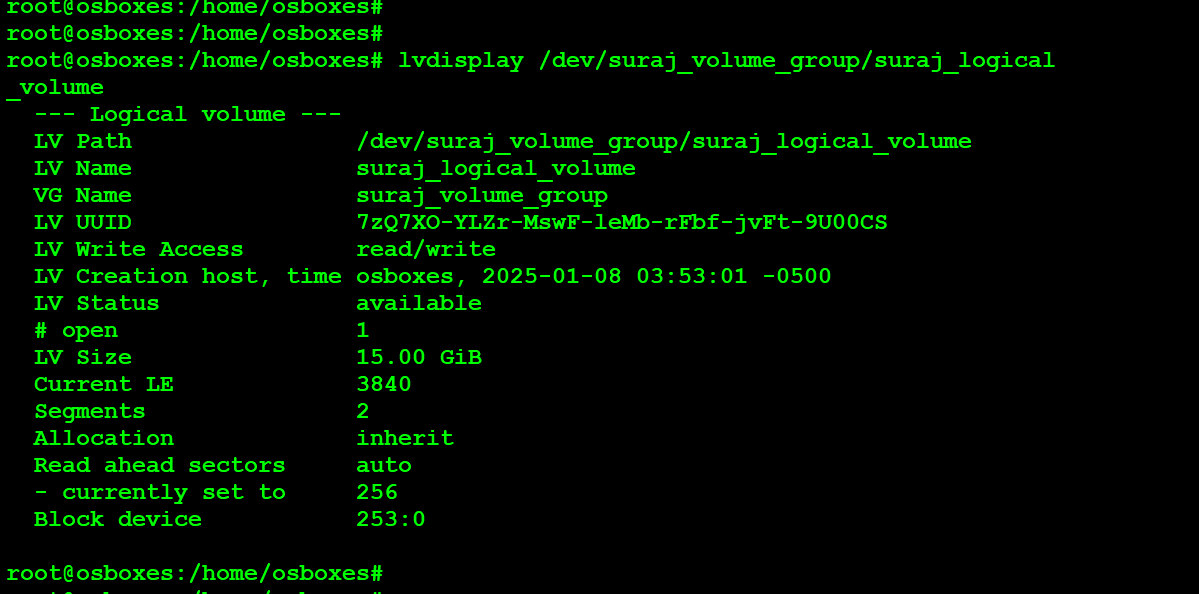
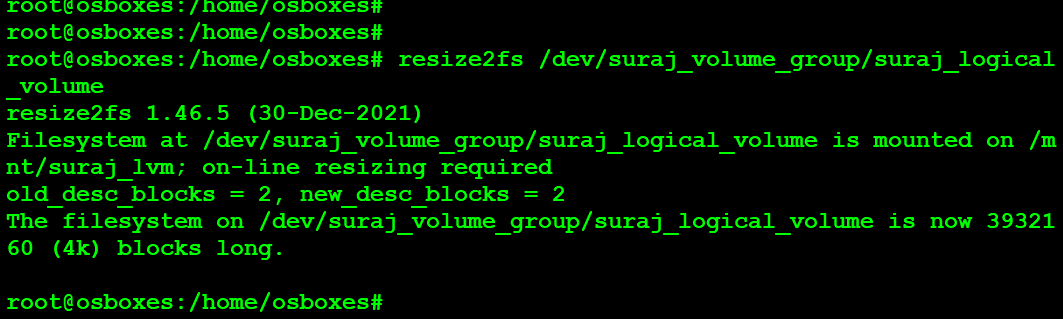
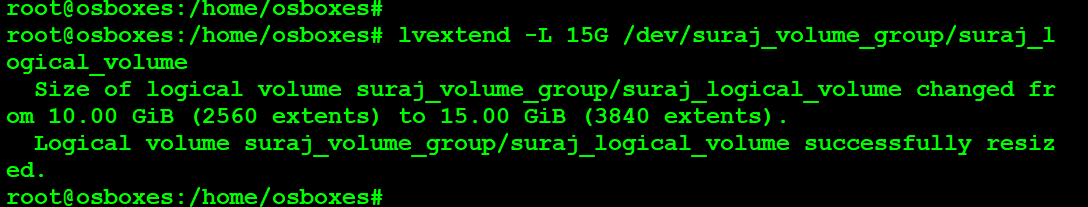
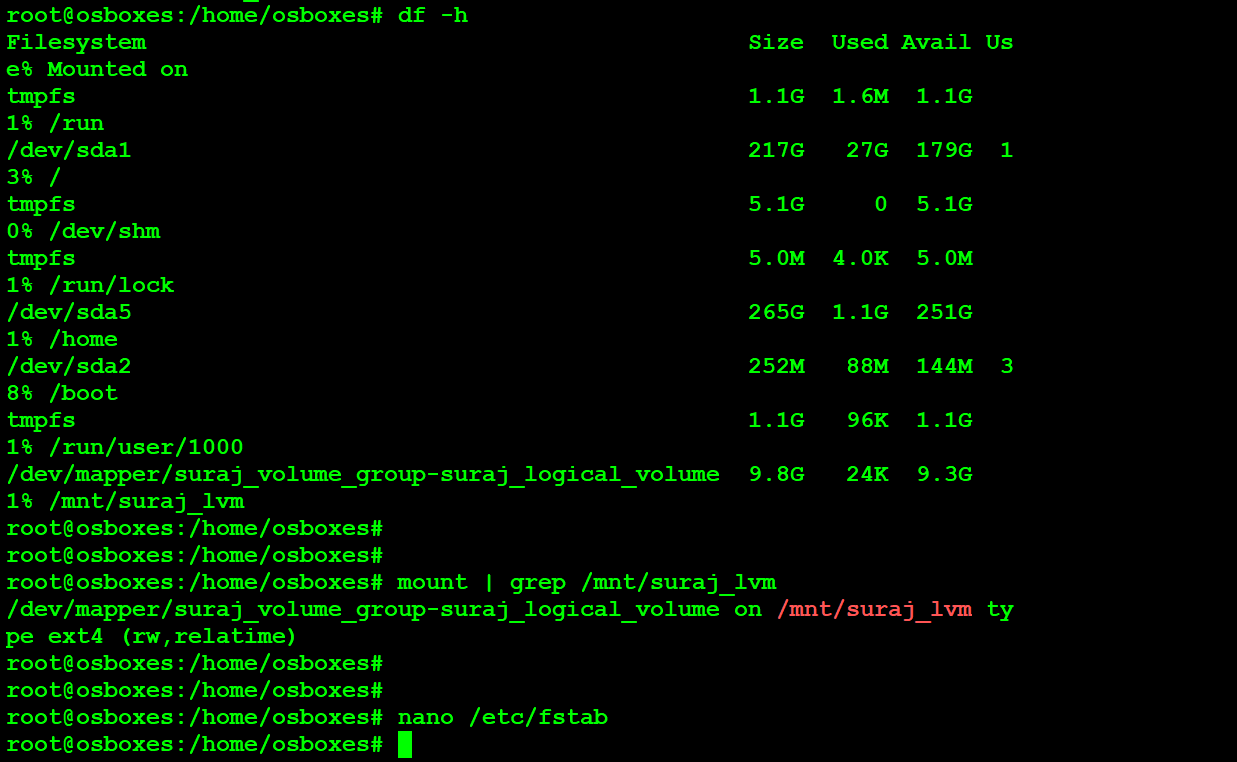
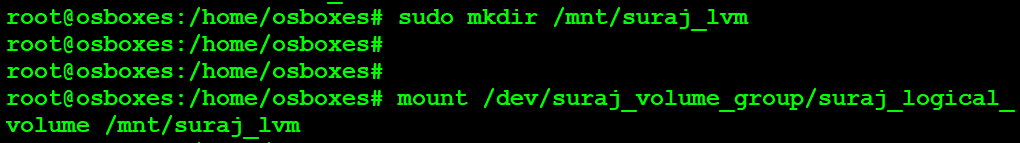
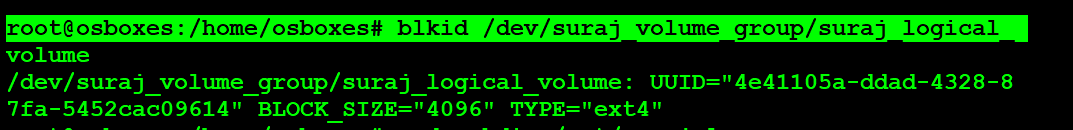
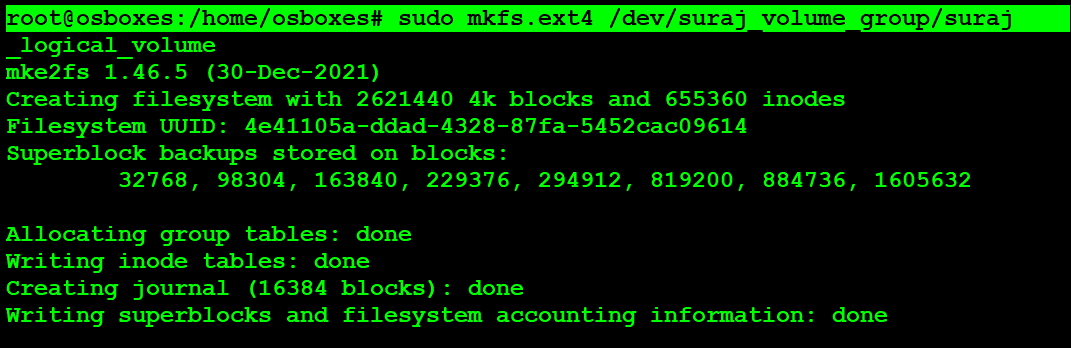
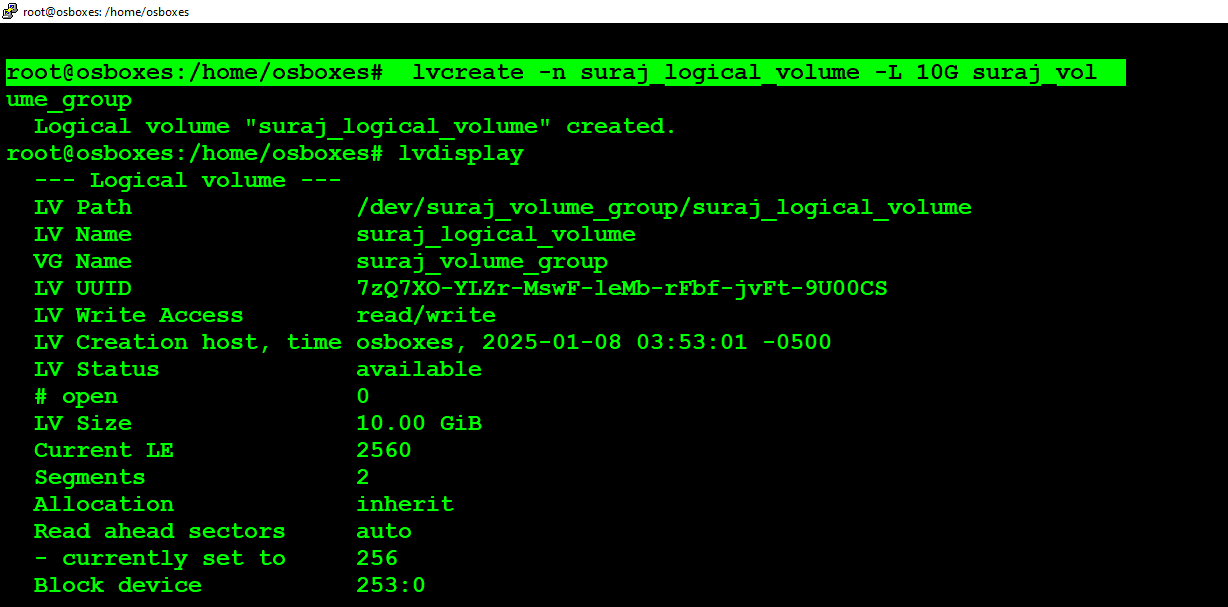
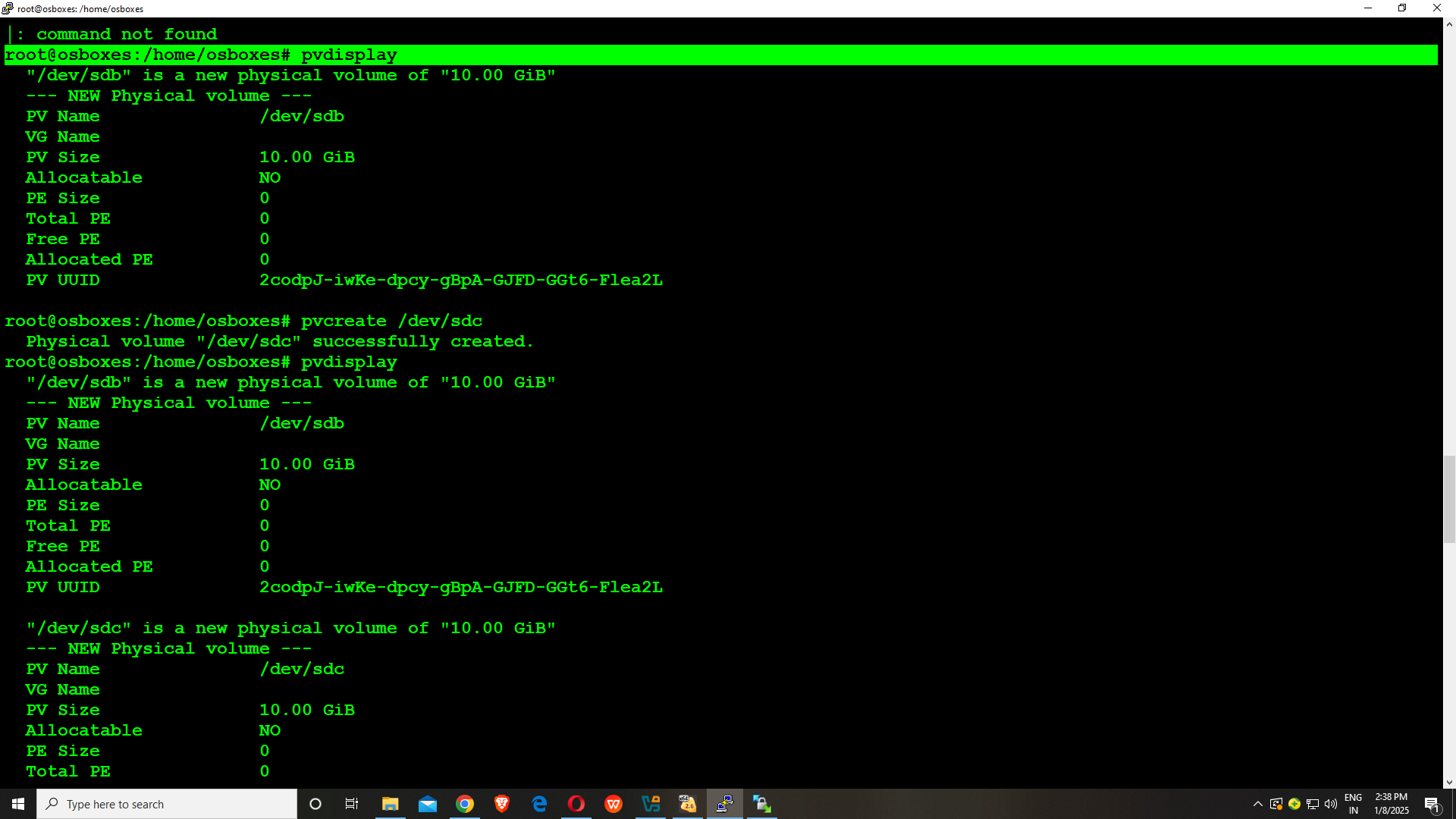
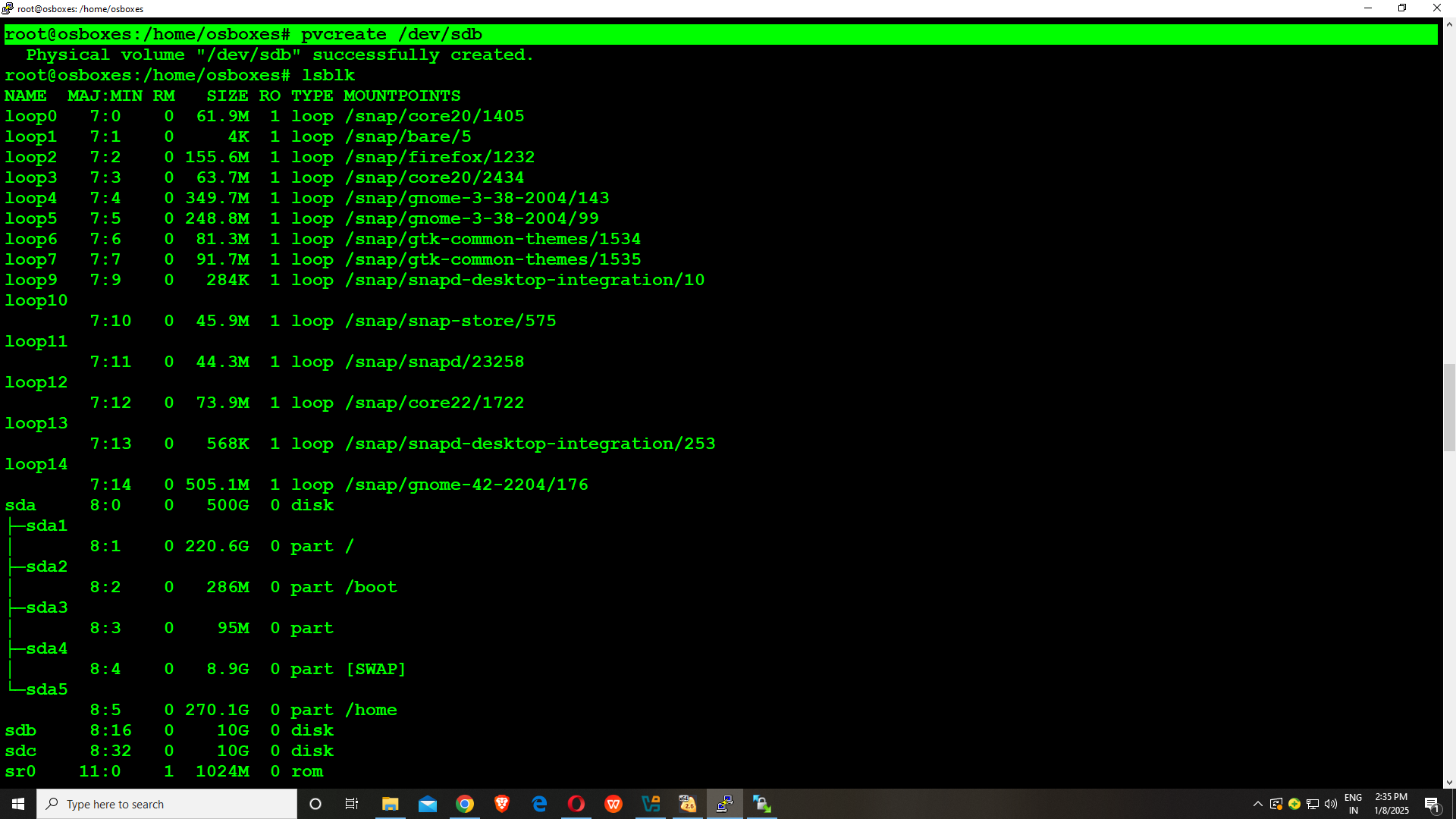
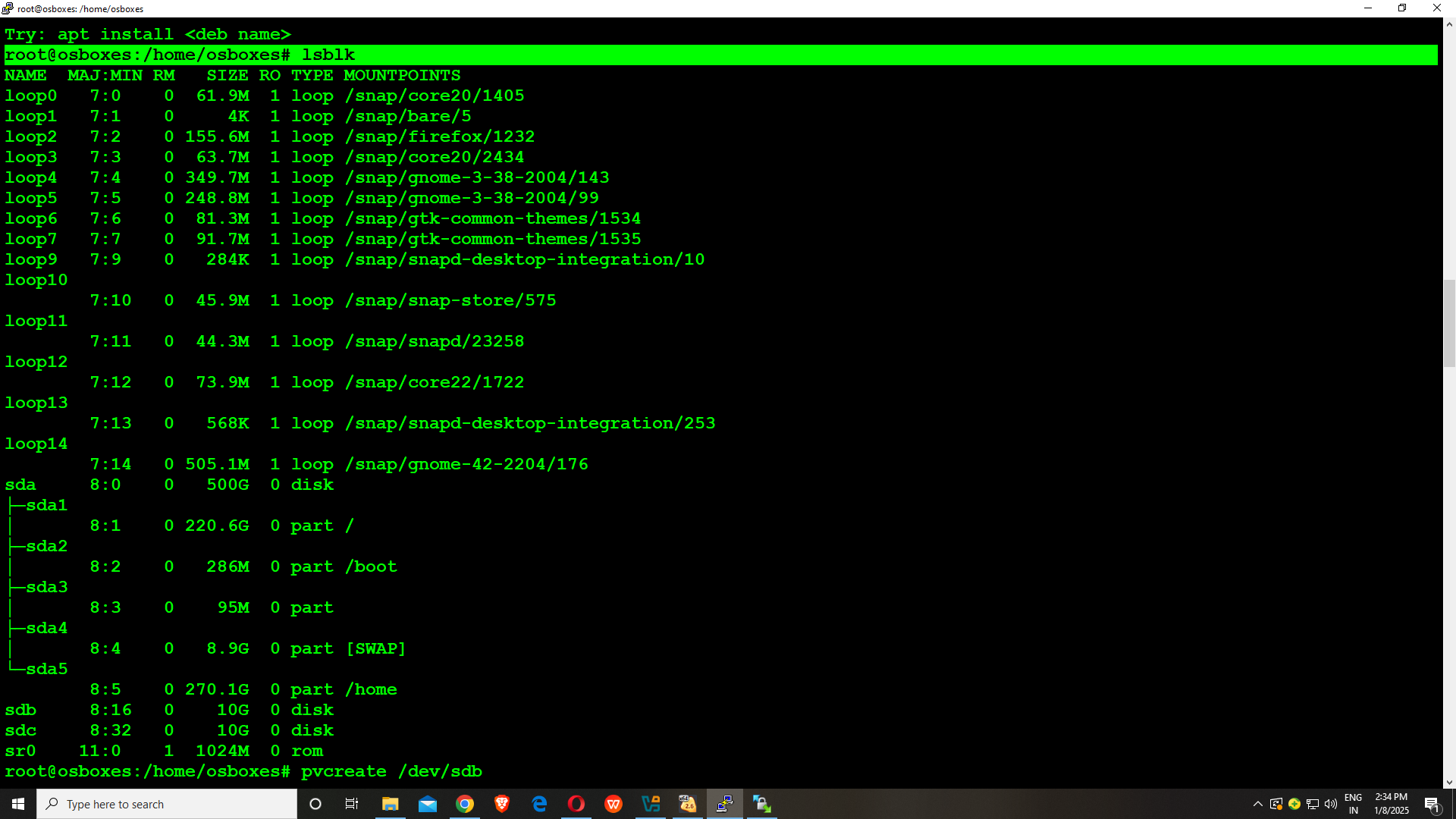
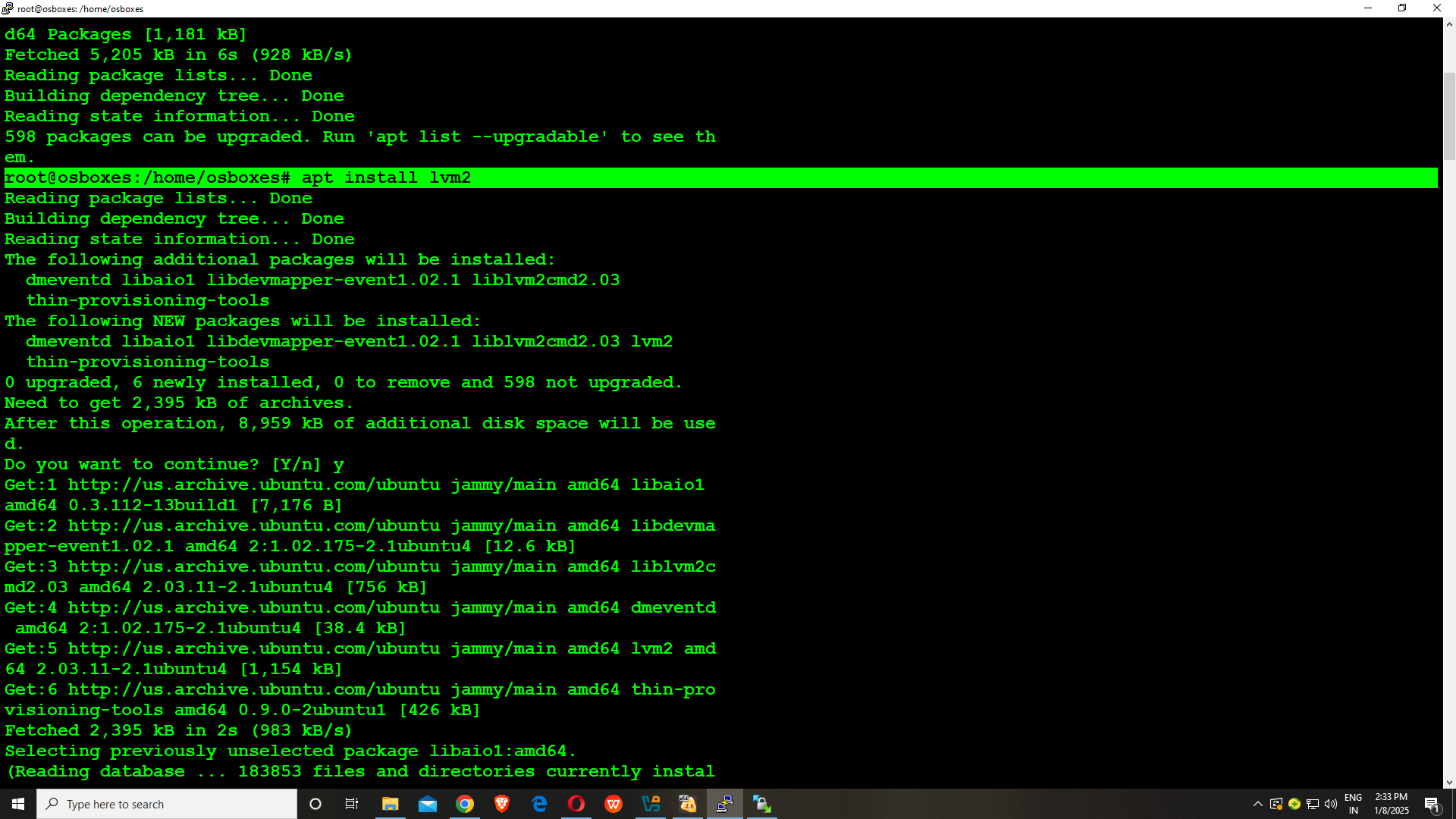
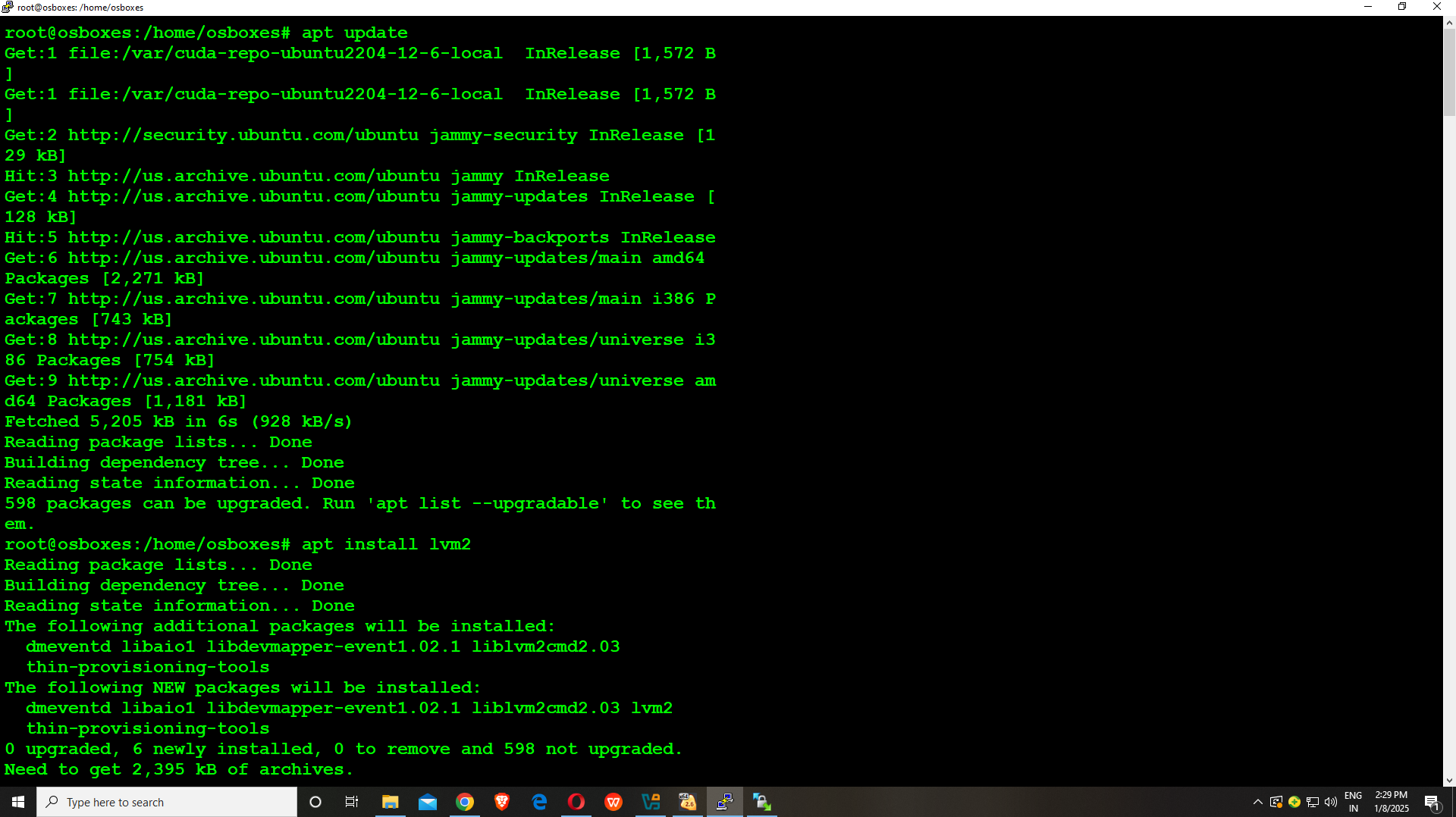
**Monitor RAID Array Health (Optional):** sudo mdadm --detail /dev/md0 # Check RAID array health

### **Additional Notes:**

* **RAID 0 does not offer redundancy**—if one drive fails, all data will be lost. It's suitable for performance, but not recommended for critical data without backup.
* You can extend the RAID array by adding more disks, but this can be a complex process and may require reconfiguration of the array.
* Always monitor the health of RAID arrays to avoid unexpected failures.



```````````````````````````````````````````````````````````LVM``````````````````````````````````````````````````



### **Layman's Summary:**

1. **Setting up RAID 0**:  
 You have two hard drives (let’s call them Disk 1 and Disk 2). RAID 0 is like combining these two drives into one big virtual drive, making it faster to read and write data. You’re using special software (mdadm) to combine them into a single "RAID" drive.

2. **Formatting the drive**:  
 After combining the two disks into RAID 0, you need to "prepare" the new drive by giving it a file system (like organizing folders on a clean desk). In this case, you're using ext4, which is a common type of file system on Linux.

3. **Making it usable**:  
 Then, you create a folder (like a drawer or a folder on your desktop) called /mnt/raid0 where this new combined drive will be stored. After that, you "mount" the RAID drive there, so your computer can start using it to store files.

4. **Checking the setup**:  
 Finally, you check if everything is working by using some commands to make sure the RAID array is properly created and that everything is ready to go. This is like double-checking your work to confirm that everything is set up correctly.

In short, you're creating a faster combined drive from two disks, preparing it to store data, and making sure it’s ready to use.

Let's go through the commands one by one to explain what each is doing:

1. **lsblk**

○ **What it does:** Lists all the available block devices (hard drives, SSDs, partitions, etc.) on your system. It shows device names (e.g., /dev/sda, /dev/sdb) along with their sizes and mount points if applicable.

○ **Why we use it:** To check the available storage devices before setting up RAID.

2. **apt update**

○ **What it does:** Updates the package index on the system, which allows you to get the latest information about available packages from your package repositories.

○ **Why we use it:** To ensure that the system has the latest package information before installing software.

3. **apt-get install mdadm -y**

○ **What it does:** Installs the mdadm tool, which is used for managing and monitoring Linux software RAID arrays.

○ **Why we use it:** mdadm is necessary to create, manage, and monitor RAID arrays on a Linux system.

4. **mdadm --create --verbose /dev/md0 --level=0 --raid-devices=2 /dev/sdb /dev/sdc**

○ **What it does:** Creates a new RAID 0 array named /dev/md0 using two devices, /dev/sdb and /dev/sdc. RAID 0 (striping) spreads data across both disks to increase performance but offers no redundancy (if one disk fails, data is lost).

○ **Why we use it:** This command sets up the RAID array with the desired configuration, specifying the RAID level (0), the number of devices (2), and the devices to include in the array.

5. **cat /proc/mdstat**

○ **What it does:** Displays the current status of all active RAID arrays on the system.

○ **Why we use it:** To verify that the RAID array was created successfully and to check its status (whether it's rebuilding, syncing, or online).

6. **mdadm --examine /dev/sdb**

○ **What it does:** Displays detailed information about the RAID metadata on the /dev/sdb disk. It checks whether /dev/sdb is part of any existing RAID array.

○ **Why we use it:** To inspect the disk and verify it has been properly recognized as part of the RAID array.

7. **mdadm --examine /dev/sdc**

○ **What it does:** Similar to the previous command, but for the /dev/sdc disk.

○ **Why we use it:** To inspect the second disk (/dev/sdc) in the RAID array and confirm it’s part of the RAID.

8. **mdadm --detail /dev/md0**

○ **What it does:** Provides detailed information about the RAID array /dev/md0, including the status, devices, RAID level, and more.

○ **Why we use it:** To check the health and status of the RAID array in more detail.

9. **mkfs.ext4 /dev/md0**

○ **What it does:** Formats the newly created RAID array (/dev/md0) with the ext4 file system.

○ **Why we use it:** We need to format the RAID array with a file system before it can be used for storing data.

10. **Press n**

○ **What it does:** During the file system creation process, this would typically be a prompt asking if you’re sure you want to proceed with formatting (or if there is a warning, like overwriting existing data).

○ **Why we use it:** To confirm or skip specific warnings during the mkfs process.

11. **Enter enter**

○ **What it does:** After pressing "n" to skip or confirm a prompt, pressing "Enter" will accept the default options during the file system setup.

○ **Why we use it:** To complete the formatting process without changing default options.

12. **mkdir /mnt/raid0**

○ **What it does:** Creates a directory /mnt/raid0 where the RAID array will be mounted.

○ **Why we use it:** To create a mount point for the RAID array so that it can be accessed from the file system.

13. **mount /dev/md0 /mnt/raid0**

○ **What it does:** Mounts the RAID array (/dev/md0) to the directory /mnt/raid0.

○ **Why we use it:** To make the RAID array accessible via the file system, so we can read/write data to it.

14. **ls -la /mnt/raid0**

○ **What it does:** Lists the contents of the /mnt/raid0 directory with detailed information (-la includes hidden files).

○ **Why we use it:** To confirm that the RAID array is mounted correctly and check the contents of the mount point (which should be empty if it’s newly formatted).

15. **mdadm --detail --scan --verbose**

○ **What it does:** Scans for all RAID arrays on the system and provides a detailed verbose output of each one.

○ **Why we use it:** To get a summary of all RAID arrays on the system and ensure the newly created array is detected and configured correctly.

### **Summary:**

In these commands, you are creating a RAID 0 array using two disks (/dev/sdb and /dev/sdc), formatting it with ext4, and mounting it to a directory (/mnt/raid0). You also verify the status and details of the RAID array using mdadm and ensure that the system recognizes and correctly configures the array.

### 

### **RAID 1**

### **RAID 1 is a data mirroring technology that keeps your data safe by copying it exactly onto two or more hard drives. It’s like having a backup of your files that’s automatically updated every time you add or change something.**

#### **Imagine it like this:**

You have **two notebooks** (hard drives), and you want to make sure all the notes you write in the first notebook are also written in the second one, so if one notebook is lost or damaged, you still have the same notes in the other notebook.

Here’s a simpler breakdown:

### **What RAID 1 Does:**

● **Duplicates data**: Every file or piece of data you save gets copied exactly to both drives (mirroring).

● **Data protection**: If one drive fails, you still have all your data on the other drive. So, RAID 1 gives you **redundancy**—a backup that happens automatically.

● **No extra space**: The downside is that you only get as much storage as the smallest drive. For example, if you have two 500GB drives, your total storage will only be 500GB because everything is duplicated.

### **Key Points of RAID 1:**

1. **Mirrored Drives**: Your data is copied (mirrored) exactly onto two drives. It’s like having an identical twin of your data.

2. **Redundancy**: If one of the drives fails, you won’t lose your data because it’s still available on the other drive.

3. **No performance boost for writing**: Because everything is written to both drives, it can actually be **slower to write** than a single drive. However, **reading** from either drive is faster, as it can pull data from whichever drive is quickest.

4. **Simple to understand**: If one drive breaks, just replace it, and the system will rebuild the data automatically from the remaining drive.

### **Why Use RAID 1?**

● **Data Security**: RAID 1 is great for anyone who wants to **protect** their data from hard drive failures. For example, if you store important documents, photos, or videos, RAID 1 makes sure you don’t lose them if something goes wrong with one of your drives.

● **Peace of Mind**: RAID 1 gives you an automatic backup. You don’t have to remember to copy files to a backup drive or cloud storage.

### **Example:**

Let’s say you have two 1TB hard drives. With RAID 1:

● When you store **500GB of data**, it will be **duplicated on both drives**. You still only have 1TB of usable space, but you have a copy of your data on each drive.

● If **one of the drives breaks**, you can just replace it, and the data will be copied back from the other drive. You don’t lose any information.

### **Disadvantages of RAID 1:**

● **Cost**: You need at least two drives for RAID 1, and since everything is duplicated, you only get half of the storage space.

● **Not a full backup solution**: While RAID 1 protects against hard drive failure, it **doesn’t protect** you from things like accidental deletion, corruption, or malware. RAID 1 isn’t a **substitute for a backup**; it’s a **redundant** storage solution.

### **Example Scenario:**

Imagine you have a business and you're storing important customer data on a computer. If the hard drive fails and you don’t have a backup, all that data could be lost. But with **RAID 1**, the same data is stored on two drives. So if one drive crashes, you still have an exact copy of all your data on the other drive. It’s like having a second, automatic backup, without you needing to do anything.

### **Conclusion:**

RAID 1 is a simple way to **protect your data** from hard drive failures by **mirroring** it across two drives. It's not a substitute for a true backup, but it’s a great way to make sure you don’t lose data if one drive fails. It’s perfect for people who want **peace of mind** and don’t mind losing a little bit of storage space for the extra protection.

To provide a comprehensive guide for implementing **RAID 1** using mdadm on Debian, including the process of verification and disk removal and addition, I'll break down each step with the necessary details. This guide will walk you through setting up the RAID 1 array, verifying the data, and simulating disk failure and rebuild scenarios to ensure the array is working properly.

### **Steps to Set Up RAID 1 with mdadm on Debian**

### **1. Install mdadm and Prerequisites**

First, ensure mdadm is installed on your Debian system:

sudo apt update

sudo apt install mdadm

### **2. Create RAID 1 Array**

To create a **RAID 1** array with two disks (let’s assume /dev/sdb and /dev/sdc):

sudo mdadm --create --verbose /dev/md1 --level=1 --raid-devices=2 /dev/sdb /dev/sdc

● /dev/md1: This will be the name of your RAID device.

● --level=1: Specifies RAID 1 (mirroring).

● --raid-devices=2: Indicates the number of devices in the RAID array.

● /dev/sdb, /dev/sdc: The two disks to use for the RAID array.

This command will create the RAID array. You can monitor the progress using:

sudo cat /proc/mdstat

### 

### **3. Save RAID Configuration to mdadm.conf**

Once the RAID array is created, save the array configuration to /etc/mdadm/mdadm.conf:

sudo mdadm --detail --scan /dev/md1 >> /etc/mdadm/mdadm.conf

This step ensures that the RAID configuration is saved and persists across reboots. Update the initramfs to include the RAID configuration:

sudo update-initramfs -u

### **4. Format the RAID Array**

Next, create a file system on the RAID device (/dev/md1):

sudo mkfs.ext4 /dev/md1

This will format the RAID device with the ext4 filesystem. You can replace ext4 with another filesystem if needed (e.g., xfs, btrfs).

### **5. Mount the RAID Array and Add a Test File**

Create a directory to mount the RAID array:

sudo mkdir /mnt/raid\_1

Now, mount the RAID array to this directory:

sudo mount /dev/md1 /mnt/raid\_1/

Create a test file to verify that the array is working properly:

echo "This is a test file for RAID 1" | sudo tee /mnt/raid\_1/testfile.txt

You can verify that the file exists by listing the contents:

ls /mnt/raid\_1

And check the contents of the file:

cat /mnt/raid\_1/testfile.txt

### **6. Verify the RAID Array**

Check the status of the RAID array:

sudo mdadm --detail /dev/md1

This will show detailed information about the RAID array, including the status of the devices and the array itself.

### **7. Remove One Disk from the RAID Array**

To simulate a disk failure, you can manually "fail" one of the disks in the array. For example, to remove /dev/sdb:

sudo mdadm --manage /dev/md1 --fail /dev/sdb

After failing the disk, the array should still work (because RAID 1 mirrors data across two disks). You can check the array status again:

sudo mdadm --detail /dev/md1

You should see that /dev/sdb is now marked as "failed" or "removed", but the RAID array should still be functioning correctly.

### **8. Verify the File is Still Accessible**

Even though one disk has been removed from the array, the data is still mirrored on the remaining disk. Verify that the test file is still accessible:

cat /mnt/raid\_1/testfile.txt

The file should still be available because RAID 1 keeps an identical copy of the data on both disks.

### **9. Reassemble the RAID Array**

If you want to reassemble the RAID array after stopping it, use the following steps:

1. Stop the RAID array:  
  
 sudo mdadm --stop /dev/md1

2. Reassemble the RAID array:  
  
 sudo mdadm --assemble /dev/md1

3. Check the RAID array details to verify that it’s healthy:  
  
 sudo mdadm --detail /dev/md1

### **10. Add a New Disk to the RAID 1 Array**

To add a new disk (let’s assume /dev/sdd) to the RAID 1 array and rebuild it:

First, add the new disk to the array:  
  
 sudo mdadm --add /dev/md1 /dev/sdd

1. The array will start to rebuild. You can monitor the rebuilding process with:  
  
 sudo cat /proc/mdstat

2. After the rebuild is complete, verify the RAID status again:  
  
 sudo mdadm --detail /dev/md1

### **11. Verify the File After Adding the New Disk**

Finally, verify that the test file is still intact on the RAID array after adding a new disk and rebuilding the array.

cat /mnt/raid\_1/testfile.txt

The file should still be present and readable, confirming that the RAID array is functioning properly and that the data is still mirrored across all disks.

### **12. Save the RAID Configuration Again (Post-Modification)**

If you’ve made changes to your RAID array (such as adding new disks), update the mdadm.conf file:

sudo mdadm --detail --scan /dev/md1 >> /etc/mdadm/mdadm.conf

Then, update the initramfs again to ensure the changes are persistent across reboots:

sudo update-initramfs -u

**Extra History Commands for Reference**

Here’s a list of additional mdadm commands you might have run as part of your process:

**Creating RAID 1**:  
  
 mdadm --create --verbose /dev/md1 --level=1 --raid-devices=2 /dev/sdb /dev/sdc

**Saving the RAID Configuration**:  
  
 mdadm --detail --scan /dev/md1 >> /etc/mdadm/mdadm.conf

**Rebuilding the RAID Array**:  
 mdadm --add /dev/md1 /dev/sdd

**Verifying RAID Status**:  
  
 mdadm --detail /dev/md1

**Stopping the RAID Array**:  
  
 mdadm --stop /dev/md1

**Reassembling the RAID Array**:  
  
 mdadm --assemble /dev/md1

### **Conclusion**

You now have a fully functional RAID 1 array on Debian, with steps for creating the array, verifying data, and simulating disk failure and rebuild. The steps above ensure that the RAID 1 setup works as expected, with data mirrored across disks, and you can recover from disk failure by adding new disks and rebuilding the array.

## 

## 

## **Complete RAID 5 Setup and Management with mdadm**

### **Prerequisites**

● **You have three virtual disks (e.g., /dev/sdb, /dev/sdc, /dev/sdd) attached to your system.**

● **Your goal is to create a RAID 5 array, mount it, and test its resilience by simulating disk failure and recovery.**

### **Step-by-Step Guide**

### **1. Install mdadm**

**First, install mdadm if it's not already installed. This tool will help you manage the RAID array.**

**apt-get install mdadm**

### **2. Create the RAID 5 Array**

**Now, create a RAID 5 array using the three disks /dev/sdb, /dev/sdc, and /dev/sdd:**

**mdadm --create --verbose /dev/md0 --level=5 --raid-devices=3 /dev/sdb /dev/sdc /dev/sdd**

**Explanation:**

● **--create: Creates a new RAID array.**

● **/dev/md0: Name of the new RAID array.**

● **--level=5: Specifies RAID 5 (with redundancy and striping).**

● **--raid-devices=3: We are using 3 disks in the array.**

● **/dev/sdb /dev/sdc /dev/sdd: The disks used for the RAID.**

### **3. Verify the RAID Array**

**To confirm that the array was created, list all block devices:**

**lsblk**

**You should see /dev/md0 as a new device.**

### **4. Save the RAID Configuration to /etc/mdadm/mdadm.conf**

**This ensures that the RAID array is automatically recognized after a reboot.**

**mdadm --detail --scan /dev/md0 >> /etc/mdadm/mdadm.conf**

### **5. Stop the RAID Array**

**If you need to stop the array (for example, for maintenance), use the following command:**

**mdadm --stop /dev/md0**

### **6. Create the Filesystem on the RAID Array**

**Format the newly created RAID array with a filesystem (commonly ext4):**

**mkfs.ext4 /dev/md0**

### 

### **7. Create a Mount Point**

**Create a directory to mount the RAID array:**

**mkdir /mnt/raid\_5**

### **8. Mount the RAID Array**

**Mount the RAID array to the newly created directory:**

**mount /dev/md0 /mnt/raid\_5/**

### **9. Verify the Mounting**

**Check that the array is mounted correctly:**

**df -h**

### **10. Create and Edit a Test File**

**Navigate to the mount point and create a test file:**

**cd /mnt/raid\_5/**

**nano test.txt**

**Add some content, e.g., "RAID 5 Array Test", save and exit (Ctrl+X, then Y).**

### **11. Shutdown the System**

**Shut down the system to simulate a reboot:**

**halt**

### **12. Reboot and Reassemble the RAID Array**

**After rebooting, reassemble the RAID array if it's not automatically assembled:**

**mdadm --assemble /dev/md0**

**If the array doesn’t assemble automatically, you can manually stop and reassemble it:**

**mdadm --stop /dev/md0**

**mdadm --assemble /dev/md0**

### **13. Mount the RAID Array Again**

**Once the RAID is assembled, mount it again:**

**mount /dev/md0 /mnt/raid\_5/**

### **14. Check if the Test File Exists**

**Check if the test file (test.txt) is still there after the reboot:**

**cd /mnt/raid\_5/**

**ls**

**cat test.txt**

**You should see the contents of the file.**

### **15. Verify RAID Array Details**

**Finally, check the detailed status of the RAID array:**

**mdadm --detail /dev/md0**

### **16. Remove One Disk (Simulate Failure)**

**To simulate a disk failure, remove one disk from the RAID array (e.g., /dev/sdb):**

**mdadm /dev/md0 --remove /dev/sdb**

**After removing the disk, the RAID will enter a degraded state, but the array will still work with the remaining disks.**

**Check the RAID status:**

**mdadm --detail /dev/md0**

**You should see that the RAID is now "degraded" and one disk is missing.**

### **17. Check the File in the Degraded RAID Array**

**Even with a degraded RAID, the data should still be accessible. Check if test.txt is still available:**

**cd /mnt/raid\_5/**

**ls**

**cat test.txt**

### **18. Re-Add the Removed Disk to the RAID Array**

**Re-add the removed disk (/dev/sdb) back to the RAID array:**

**mdadm /dev/md0 --add /dev/sdb**

### **19. Monitor the RAID Rebuilding Process**

**RAID 5 will start rebuilding the array to restore redundancy. Monitor the rebuild process:**

**watch cat /proc/mdstat**

**The process may take some time, depending on the size of the disks and array.**

### **20. Verify the File After Rebuild**

**Once the rebuild completes, verify that the file (test.txt) is still intact:**

**cd /mnt/raid\_5/**

**ls**

**cat test.txt**

**21. Final Verification**

**Ensure that the RAID array is clean and all disks are active:**

**mdadm --detail /dev/md0**

**Verify the filesystem is still mounted:**

**mount | grep /dev/md0**

**Check that test.txt is accessible:**

**cd /mnt/raid\_5/**

**ls**

**cat test.txt**

## **Key Points and Commands Summary**

**Create RAID 5 Array:  
 mdadm --create --verbose /dev/md0 --level=5 --raid-devices=3 /dev/sdb /dev/sdc /dev/sdd**

**Save RAID Configuration to /etc/mdadm/mdadm.conf:  
  
 mdadm --detail --scan /dev/md0 >> /etc/mdadm/mdadm.conf**

**Stop the RAID Array:  
  
 mdadm --stop /dev/md0**

**Create Filesystem:  
  
 mkfs.ext4 /dev/md0**

**Mount the Array:  
  
 mount /dev/md0 /mnt/raid\_5/**

**Check RAID Status:  
  
 mdadm --detail /dev/md0**

**Remove a Disk:  
  
 mdadm /dev/md0 --remove /dev/sdb**

**Re-add a Disk:  
  
 mdadm /dev/md0 --add /dev/sdb**

**Monitor RAID Rebuild:  
  
 watch cat /proc/mdstat**

### **Important Notes:**

● **RAID 5 can tolerate the failure of one disk, but performance will be degraded until the array is rebuilt.**

● **Rebuilding the array can take time, depending on the disk size and system performance.**

● **Data integrity should remain intact during the rebuild process, but always monitor the rebuild to ensure it completes successfully.**

A hot spare in RAID (Redundant Array of Independent Disks) refers to a standby hard drive that is not actively used in the array but is ready to automatically take the place of a failed drive. When one of the active drives in the RAID array fails, the hot spare will automatically be incorporated into the array to rebuild the lost data, minimizing downtime and maintaining redundancy without requiring manual intervention.

### **Key Points about Hot Spares:**

1. Automatic Failover: When a drive in the RAID array fails, the hot spare will be "hot-swapped" into the array without requiring any downtime or manual action. This helps to maintain redundancy and performance during the rebuild process.

2. Types of RAID Configurations Using Hot Spares:

○ RAID 1 (Mirroring): A hot spare can take the place of a failed drive and will rebuild the mirror automatically.

○ RAID 5 (Striping with Parity): If a single drive fails, a hot spare can take over to rebuild the array using the parity data from the remaining drives.

○ RAID 6 (Dual Parity): Similarly to RAID 5, a hot spare can be used to rebuild data if one or two drives fail.

3. Hot Spare vs. Cold Spare:

○ Hot Spare: A hot spare is already powered on, connected, and ready to be used immediately if a failure occurs.

○ Cold Spare: A cold spare is a spare drive that is kept in storage and must be manually swapped into the array when a failure occurs.

4. Single vs. Multiple Hot Spares: Some systems allow more than one hot spare, which can be used in larger arrays for additional protection.

5. Rebuild Process: Once the failed drive is replaced with the hot spare, the RAID controller will begin the rebuild process, copying the data from the remaining operational drives (and parity information, if applicable) to the new drive to restore redundancy.

6. Drive Health Monitoring: Hot spares are particularly useful in environments where drive failures might occur unexpectedly, providing a buffer before manual intervention is needed.

### **Advantages:**

● Reduced Downtime: No need for a system administrator to manually replace a failed drive.

● Improved Data Availability: Immediate rebuild process can restore the array's redundancy quickly.

● Increased System Reliability: Hot spares help maintain RAID array health without requiring the array to be offline.

### **Disadvantages:**

● Cost and Space: A hot spare requires having an extra drive on hand, which increases the cost and space requirements for the storage system.

● Not a Full Backup: While a hot spare helps with redundancy, it is not a substitute for regular backups of critical data.

In summary, a hot spare is a proactive measure in RAID setups to ensure rapid recovery from disk failures, improving the overall reliability and uptime of the storage system.

A **hot spare** in RAID is a standby drive that is not actively used but is ready to automatically replace a failed drive. When a drive in the RAID array fails, the hot spare is instantly activated to rebuild data, ensuring minimal downtime and maintaining redundancy. It’s different from a cold spare, which must be manually swapped in. Hot spares are commonly used in RAID 1, RAID 5, and RAID 6 setups to enhance data protection and system reliability.