In Linux, disk management and volume management are two distinct concepts that involve the organization, allocation, and control of storage resources. Let's explore each concept:

**Disk Management**

Disk management in Linux refers to the process of handling physical storage devices, such as hard drives or solid-state drives (SSDs). This involves tasks like partitioning, formatting, and managing file systems on these storage devices. Here are some key components of disk management:

* Partitioning: Dividing a physical disk into separate logical sections called partitions. Each partition can be treated as an independent unit with its own file system.
* Formatting: Creating a file system on a partition, which allows the operating system to organize and store data on that partition.
* Device Naming: Identifying and managing storage devices using device names like `/dev/sda` for the first SATA drive or `/dev/nvme0n1` for the first NVMe drive.
* Mounting: Attaching a partition or disk to a specific directory in the file system hierarchy, making its contents accessible to the user and applications.

Common tools for disk management in Linux include `fdisk`, `parted`, and `gparted`.

**Volume Management**

Volume management, on the other hand, involves handling logical storage structures that may span multiple physical disks. The primary purpose is to abstract the underlying physical storage and provide a unified view of storage resources to the operating system and applications. Key aspects of volume management include:

* Volume Groups: Combining multiple physical volumes (disks or partitions) into a single logical volume group. This allows for better utilization and management of storage space.
* Logical Volumes: Slicing the volume group into logical volumes. Logical volumes are similar to partitions but can span multiple physical volumes, providing flexibility in terms of size and management.
* File Systems: Creating file systems on logical volumes, similar to the file system creation on partitions.
* Online Resize: The ability to resize logical volumes while the file system is mounted and in use.

Common tools for volume management in Linux include `LVM` (Logical Volume Manager) and tools like `lvcreate`, `lvextend`, and `lvresize` for managing logical volumes.

In summary, disk management deals with the physical storage devices, while volume management provides a logical layer on top of physical storage, offering more flexibility and ease of management, especially in the context of dynamic storage requirements and scalability.

**Logical Volume Management**

A logical volume manager (LVM) is a type of storage virtualization that combines multiple hard drives/or partitions into a single volume group (VG), which then can be divided into Logical Volumes (LV), or can be used as single large volume.

An LV is similar to a partition on a physical disk. Multiple LVs can be carved out of a VG, which itself is a storage pool consisting of multiple PVs. The aggregate storage capacity of the PVs is abstracted by the VG in order to create logical units of storage called LVs.

# lsblk 🡪 Lists information about block devices, including partitions.

# blkid 🡪 Displays information about block devices, including UUIDs and file system types.

To create Physical volumes

# pvcreate /dev/sd’**x**’ 🡪 Pvcreate\_path\_of\_the\_disk. By default, all disks/drives will be in the directory dev and hard disks are named as sda, sdb, sdc and so on.

To create multiple physical volumes pvcreate /dev/sd{a,b}

# pvs 🡪 used for displaying detailed information about physical volumes. Like list of physical volumes, total space, free space.

# pvs -v 🡪 extension of pvs displays more detailed information about physical volumes. Like physical volume(pv) name, volume group(vg) name, format of pv, attributes (attr) of pv, total size, free space, Pv UUID etc.

The pvscan command scans all available devices for LVM physical volumes and updates the LVM cache. Like it shows which hard drives are converted to physical volumes. Used to refresh the LVM metadata cache after changes have been made to the configuration.

# pvdisplay 🡪 displays detailed information about a physical volume in the Logical Volume Management (LVM) setup. It provides an extensive overview of the characteristics and attributes associated with a specific physical volume.

To create volume group

# vgcreate group\_name dev/sdb dev/sdc 🡪 vgcreate desired-group-name and path of the physical volumes you want to group

# vgs 🡪 Display information about volume groups. Like name of the volume group, the number of physical volumes it spans, the total size, the number of logical volumes it contains, and more.

# vgs -v 🡪 extension of vgs displays more detailed information about volume groups.

To create Logical volumes

# lvcreate -L Size -n LogicalVolumeName VolumeGroupName 🡪 lv create -L 5G -n lv\_1 vg\_1

# lvs 🡪 display information about logical volumes. Like name of the logical volume, the volume group to which it belongs, its size, the number of physical extents allocated, and more.

After creating the logical volume, you typically need to create a file system on it and mount it.

# mkfs.ext4 /dev/volume\_group/logical\_volume 🡪

A file system is a method for storing and organizing data on a computer. Ext4 is the default file system of Red Hat Enterprise Linux 6, and can support files and file systems up to 16 terabytes in size.

Now run # partprobe 🡪 Not exactly a command, but a utility that informs the operating system about partition table changes without requiring a reboot.

mount command is used to attach a logical volume or a partition to a specific directory in the file system hierarchy.

# mount /dev/VolumeGroupName/LogicalVolumeName /directory/🡪 ex: mount /dev/vg\_data/my\_lv

# df -h 🡪 used to display information about disk space usage on mounted file systems. This command will show you a human-readable summary of disk space usage for all mounted file systems.

To write to a document on a mounted logical volume (LVM), Change your working directory to the mount point where your logical volume is mounted. Use a text editor (e.g., nano, vim, gedit) to create or edit a document.

The lvextend command is used to extend the size of a logical volume.

# lvextned -L+space\_to\_be\_added /dev/vg\_name/lv\_name 🡪 Ex: lvextend -L +5G /dev/vgdata/mylv

The lvreduce command is used to reduce the size of a logical volume.

# lvreduce -L+space\_to\_be\_reduced /dev/vg\_name/lv\_name 🡪 Ex: lvextend -L +2G /dev/vgdata/mylv

Snapshot

A snapshot is a feature that allows you to create a point-in-time copy. Snapshots are useful for various purposes, such as backups, data consistency, and testing, without disrupting ongoing operations on the original logical volume.

Snapshot is different from backup.

To create a snapshot, you use the lvcreate command

# lvcreate -s -n snapshot\_name size /dev/path/volume\_group

We can remove snapshot by lvremove /dev/vg\_name/snapshot\_name

Purpose of Snapshots:

Backup and Recovery: Snapshots provide a way to create a consistent backup of a logical volume without interrupting regular read and write operations.

Testing and Development: Snapshots allow you to create a copy of a logical volume for testing purposes, such as software development or system upgrades, without affecting the original data.

* If you've lost data on a logical volume and you have a snapshot available, you can potentially use the snapshot to retrieve the lost data. Create a mount point and mount the snapshot read-only.

# mount /dev/mapper/vg\_name/snapshot\_name/directory (where we want to mount)

* We can recover lost data by merging a snapshot directly into the logical volume (LV) where data was lost. Before merging, make sure the original LV is unmounted to avoid potential conflicts.

To unmount #umount /dev/mapper/vg\_name/snapshot\_name/directory

Use the lvconvert command to merge the changes from the snapshot back into the original LV

# lvconvert –merge /dev/vg\_name/snapshot\_name

After successfully merging, you should remove the snapshot to free up space

# lvremove –merge /dev/vg\_name/snapshot\_name

**Redundant Array of Independent Drives or Redundant Array Inexpensive Disks (RAID)**

RAID is a storage technology in which multiple disks are combined to act as a single storage unit in order to achieve better performance and/or reliability compared to single disks.

RAID works by storing the data on multiple disks, allowing balanced input/output or I/O operations to boost its performance. Since RAID makes use of multiple disks, it can increase the Mean Time Between Failures (MTBF) and improve fault tolerance by storing data redundantly.

There are two possible RAID approaches: hardware RAID and software RAID.

Different RAID levels are

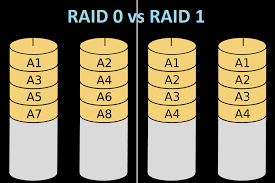
1. RAID0 = Striping
2. RAID1 = Mirroring
3. RAID5 = Single Disk Distributed Parity (striping with parity)
4. RAID6 = Double Disk Distributed Parity (striping with double parity)
5. RAID10 = Combine of Mirror & Stripe. (Nested RAID)

**RAID 0 (or) Striping**

In Raid 0 (Striping) the data will be written to disk using shared method. Half of the content will be in one disk and another half will be written to other disk.

This means the data being written to the array is broken down into strips and written across the member disks of the array.

* No redundancy; purely focused on improving performance through parallel data access.
* Not suitable for applications where data integrity and fault tolerance are critical.



**RAID 1 (or) Mirroring**

In RAID 1, the data is mirrored between disks (two or more). As a result, the data is written to each of the group's drives. In other words, each disk contains an exact copy of the same data.

* Provides high data redundancy; if one disk fails, the data is still available from the mirrored copy.
* Efficient for applications requiring data integrity and fault tolerance.

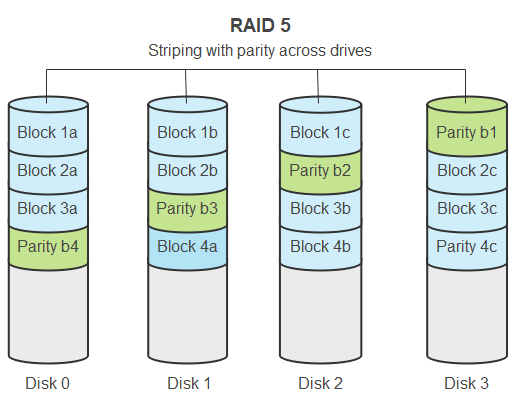
**RAID 5 (or) Distributed Parity**

RAID 5 uses techniques from both RAID 0 and RAID 1 in its setup.

It strips data across devices but also ensures that the striped data is mirrored across the array. It checks the parity information using mathematical algorithms.

Parity: Parity is a value used for retrieving data from the other drives in which it was kept as a copy of the original data in case of storage media failure. Parity is done by performing XOR (Exclusive OR) operation on the data to be stored. For a RAID protection with 3 Disks, only 2 disks will store parity.

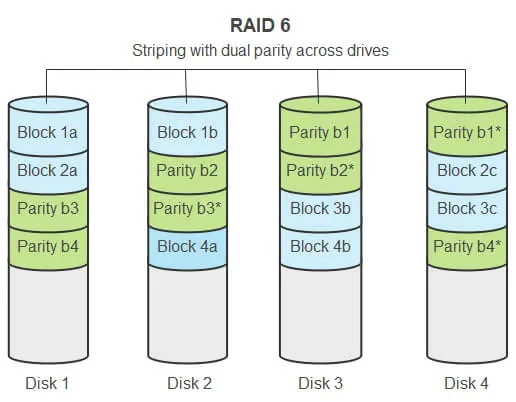
* Balances performance and redundancy; can withstand the failure of one disk.
* Suitable for general-purpose storage where both performance and data integrity are important.



**RAID 6 Two Parity Distributed Disk**

Similar to RAID 5 but with two sets of parity information.

* Improved fault tolerance; can withstand the failure of two disks.
* Used in scenarios where high fault tolerance is crucial, such as large storage arrays.



**RAID 10 (or) Mirror & Stripe**

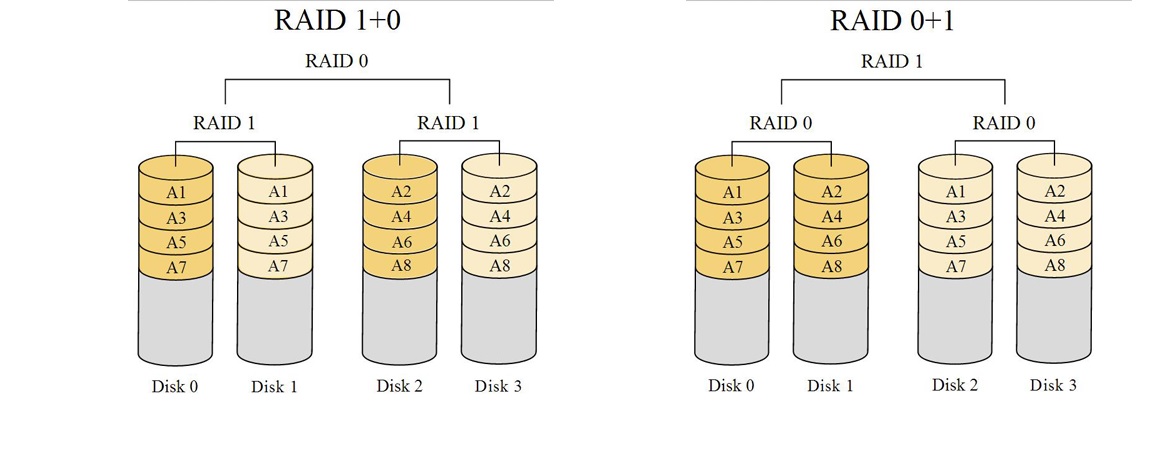
RAID 10 can be called as 1+0 or 0+1. This will do both works of Mirror & Striping.

This can be described in two ways:

Nested RAID 1+0: Mirror will be first and stripe will be the second in RAID 10

mdam's RAID 0+1: Stripe will be the first and mirror will be the second in RAID 01.

* Offers both performance and high data redundancy.
* Well-suited for environments that require a balance of performance and redundancy.



**RAID 0 Configuration**

🡪Requires at least two hard disks for the RAID 0 configuration to work.

🡪Choose disks with the same brand, type, and sizes when opting for RAID 0.

🡪 Have a backup of your data because hard disks will be erased during the RAID setup.

1. Verify Disk Devices:

First, make sure you have the appropriate disk devices available. Use commands like lsblk, fdisk -l, or parted -l to list available disks on your system.

2. Install mdadm:

mdadm is a tool for managing Linux software RAID devices.

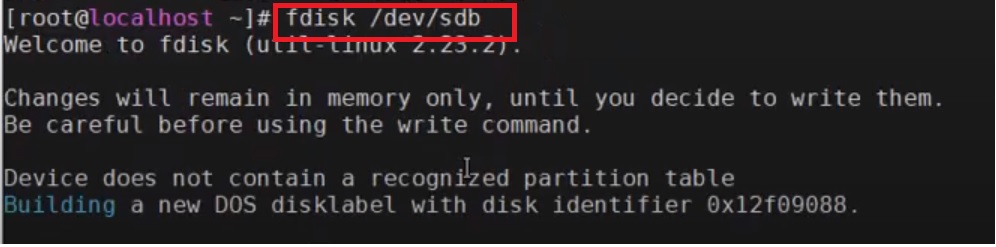
# sudo apt update

# sudo apt install mdadm

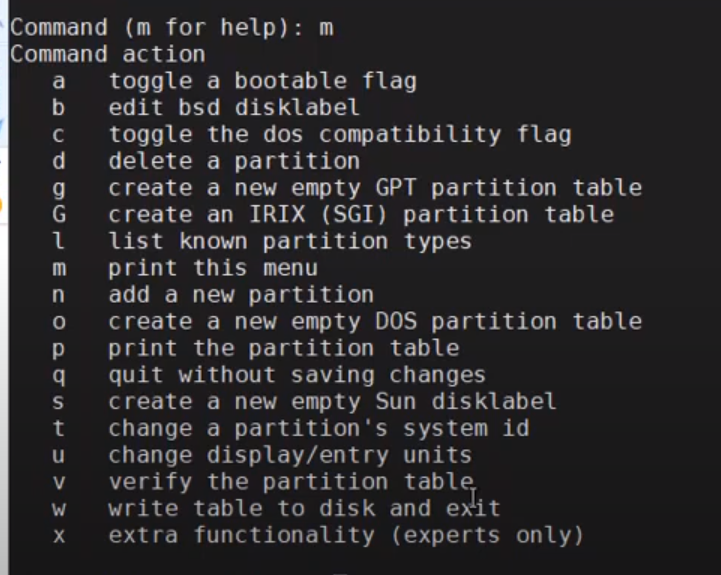
(mdadm🡪 Multiple Disk And Device Management)

3.Drive partitioning for RAID

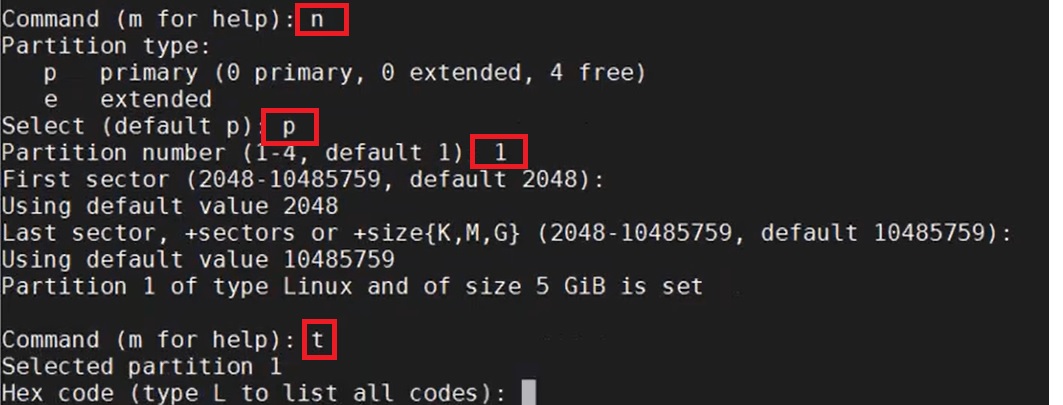
Use the command # fdisk /dev/sdb



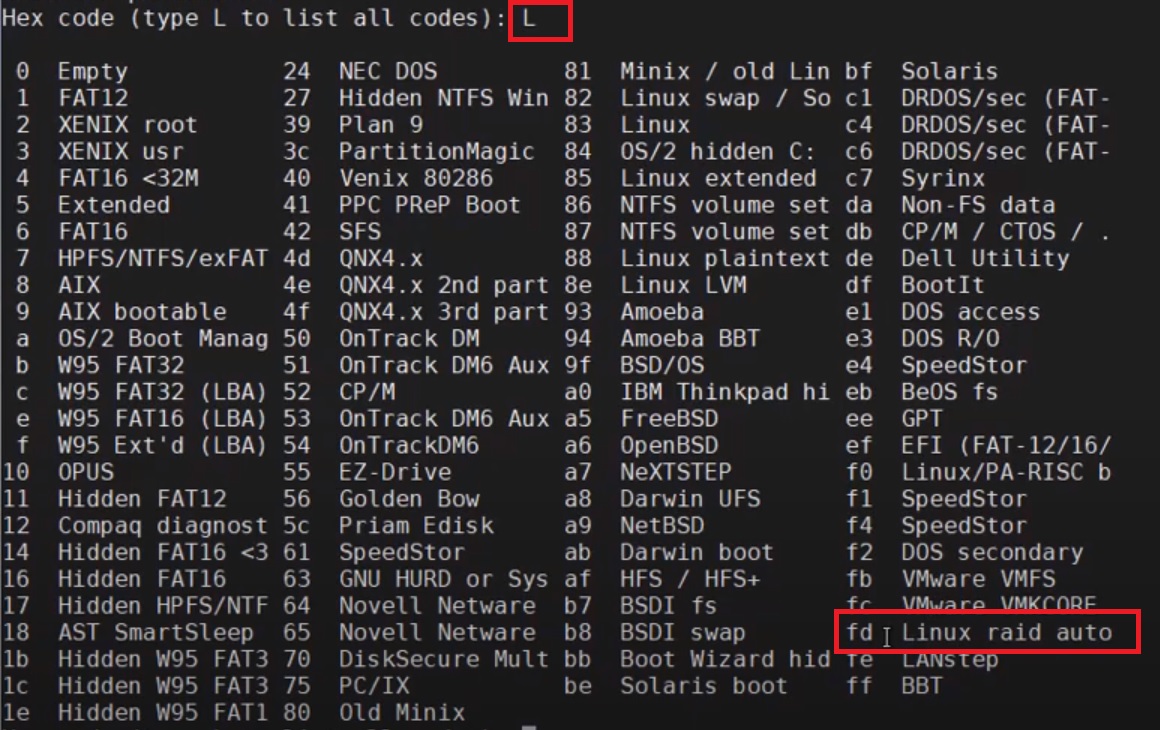
* Press ‘m‘ for help options.



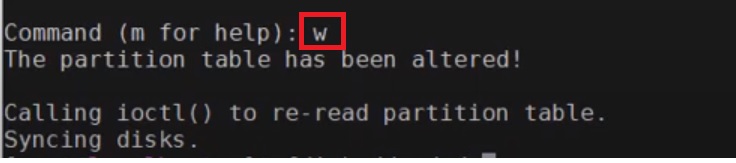
* Press ‘n’ for creating new partition
* Then choose ‘P‘ for Primary partition.
* Next select the partition number as 1.
* Type ‘t‘ to choose the partitions.



* Press ‘L‘ to list all available types. (in hex code)
* Choose ‘fd‘ for Linux raid auto and press Enter to apply.

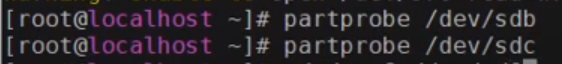


* Use ‘w‘ to write the changes.



* Now we are do the same for **sdc**

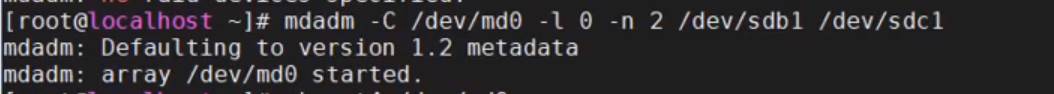
Now run the commands # partprobe /dev/sdb #partprobe /dev/sdc



informs the operating system about partition table changes without requiring a reboot

4. Create RAID 0 Array:

#mdadm -C /dev/md0 -l 0 -n 2 /dev/sdb1 dev/sdc1



/dev/md0: This is the name of the RAID device.

-l 0: Specifies RAID level 0 (striping).

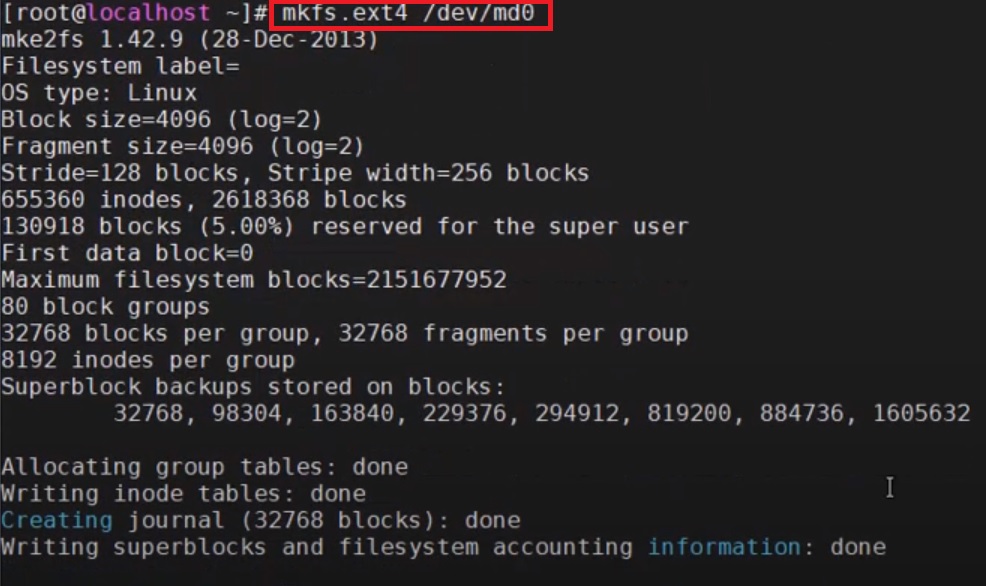
-n 2: Indicates the number of disks in the array.

/dev/sdb1 /dev/sdc1: Specify the disks you want to use in the RAID array.

5. Create File System:

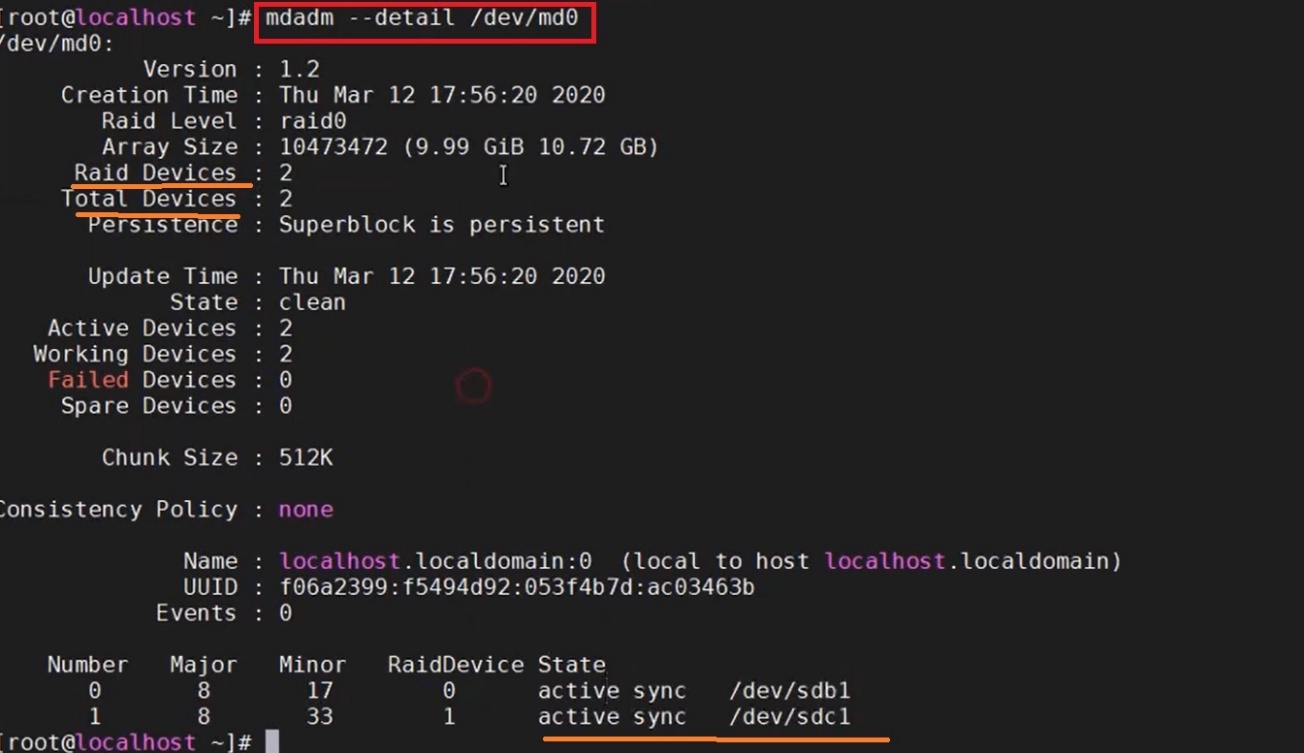
Now, we need to create a file system on the RAID device. We use mkfs to create an ext4 file system.

# mkfs.ext4 /dev/md0



Optional: we can check the details with the command

# mdadm –detail /dev/md0

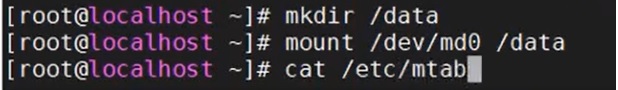


6. Mount the RAID Array:

Create a mount point and mount the RAID array:

# mkdir /data (data directory is created)

# mount /dev/md0 /data (RAID is mounted to data directory)

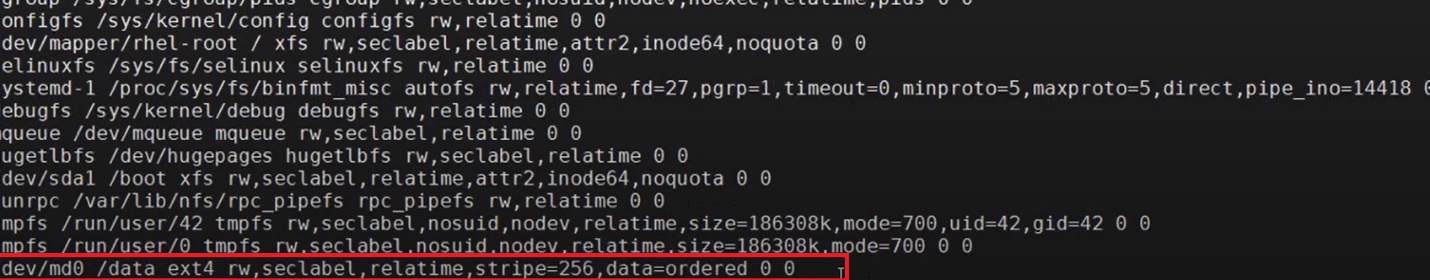


7. Update /etc/fstab for Automatic Mounting:

Add an entry to the /etc/fstab file to ensure the RAID array is mounted at boot:

The /etc/mtab or /proc/mounts will show the mount points of the RAID devices or partitions once they are mounted. When you create a RAID array using tools like mdadm and then mount it, the information about the mounted RAID devices will be reflected in this file.

# cat /etc/mtab for permanent mounting copy the last line in /etc/mtab



The "/etc/fstab" file in Linux is an important configuration file that contains information about disk drives and partitions.

# vim /etc/fstab paste that line copied from /etc/mtab and save it here.

We can check by df -h command

Thus RAID is created and we can move/create files/data into the directory created by us ‘/data’