

**INTERDISCIPLINARY PROJECT REPORT**  
**at**  
**Sathyabama Institute of Science and Technology**  
**(Deemed to be University)**

Submitted in partial fulfillment of the requirements for the award of  
Bachelor of Engineering Degree in Computer Science and Engineering

By  
**VISHWAJEET ANAND**  
(Reg. No. 40731127)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**SCHOOL OF COMPUTING**  
**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**  
**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**  
**CHENNAI – 600119, TAMILNADU**

**APRIL 2023**



# SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 600119

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)



---

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **VISHWAJEET ANAND (40731127)** who carried out the project entitled "**Managing LVM on VDO (LINUX)**" under the supervision of **Dr. Veena K, M.E., Ph.D.** from **FEB 2023** to **APRIL 2023**.

Internal Guide

**Dr. Veena K, M.E., Ph.D.**

Head of the Department

**Dr. S. Vigneshwari, M.E., Ph.D.**

Submitted for Viva voce Examination held on 17-04-2023

Internal Examiner

External Examiner

## **DECLARATION**

I **VISHWAJEET ANAND** hereby declare that the Project Report entitled “**Managing LVM on VDO**”. Done by me under the guidance of **Dr. VEENA K, M.E., PhD.** at **ADVANTAGE PRO (PRIDE)** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

**DATE: 17-04-2023**

**PLACE: CHENNAI**

A handwritten signature in blue ink, appearing to read 'Balbir', with a horizontal line underneath.

**SIGNATURE OF THE CANDIDATE**

## **ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing , **Dr.S.Vigneshwari M.E., Ph.D., and Dr.L.Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Veena K, M.E., PhD.** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

## TRAINING CERTIFICATE



# Red Hat

# Certificate of Participation

Vishwajeet Anand

has attended and successfully completed

Red Hat Certified System Administrator

ONLINE course

at Sathyabama Institute of Science and Technology

held from 17th Feb 2023 to 31st March 2023

in association with



Authorised Signatory

S.No : 2023/ADV/12512

Date : 20th April 2023

## **ABSTRACT**

The goal of this project is to explore the use of Logical Volume Manager (LVM) and Virtual Data Optimizer (VDO) in Linux storage management. LVM allows for flexible and efficient management of storage by creating logical volumes on top of physical volumes, while VDO enables space-saving compression and de-duplication of data. The project will investigate the integration of LVM and VDO, specifically the process of creating logical volumes on top of VDO devices and the benefits of this approach. The project will involve practical implementation and experimentation with LVM and VDO commands and configurations. The expected outcome of the project is a better understanding of the use of LVM and VDO together to create a more efficient and flexible storage solution in Linux environments.

**Keywords:** Logical Volume Manager (LVM), Virtual Data Optimizer (VDO).

| <b>CHAPTER<br/>No.</b> | <b>TITLE</b>                                                                                                                                                                                                                                                                                                                                                  | <b>PAGE<br/>No.</b>                                                                       |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| 1.                     | <b>ABSTRACT</b><br><b>LIST OF FIGURES</b><br><b>LIST OF ABBREVIATIONS</b><br><b>INTRODUCTION</b>                                                                                                                                                                                                                                                              | (i)<br>(ii)<br>(iii)<br><b>1</b>                                                          |
| 2.                     | <b>AIM AND SCOPE OF THE PRESENT INVESTIGATION</b><br>2.1 Aim<br>2.2 Scope                                                                                                                                                                                                                                                                                     | <b>2</b>                                                                                  |
| 3.                     | <b>MATERIAL AND METHODS</b><br>3.1 Basics of LVM on VDO<br>3.2 Commands used                                                                                                                                                                                                                                                                                  | <b>3</b>                                                                                  |
| 4.                     | <b>IMPLEMENTATION OF THE PROJECT</b><br>4.1 Problem with standard partitions<br>4.2 Solutions with the LVM<br>4.3 Partitioning<br>4.3.1 Partition Table<br>4.3.2 Types of Partitions<br>4.3.3 Linux File System<br>4.4 LVM Terminologies<br>4.4.1 Physical Volume<br>4.4.2 Volume Group<br>4.4.3 Logical Volume<br>4.4.4 Commands<br>4.5 VDO<br>4.6 Execution | <b>4-23</b><br>4<br>4<br>5-7<br>5<br>6<br>7<br>8-10<br>8<br>8<br>8<br>9-10<br>10-11<br>12 |
| 5.                     | <b>SUMMARY AND COLCLUSION</b><br><br><b>REFERENCES</b><br><br><b>SOURCE CODE</b>                                                                                                                                                                                                                                                                              | <b>24</b><br><br><b>25</b><br><br><b>25-26</b>                                            |

## LIST OF FIGURES

| FIGURE NO. | FIGURE NAME            | PAGE NO. |
|------------|------------------------|----------|
| 1.         | Managing LVM on VDO    | 1        |
| 2.         | Partitions             | 4        |
| 3.         | Use case of LVM        | 4        |
| 4.         | MBR Structure          | 5        |
| 5.         | Overview of Partitions | 6        |
| 6.         | FHS on Linux           | 7        |
| 7.         | Overview of LVM        | 9        |

## LIST OF ABBREVIATIONS

| ABBREVIATION | EXPANSION              |
|--------------|------------------------|
| LVM          | Logical Volume Manager |
| VDO          | Virtual Disk Optimizer |
| MBR          | Master Boot Record     |
| FHS          | File Hierarchy System  |
| PV           | Physical Volume        |
| LV           | Logical Volume         |
| VG           | Volume Group           |

## CHAPTER 1: INTRODUCTION

Managing storage in Linux environments is critical, and Logical Volume Manager (LVM) and Virtual Data Optimizer (VDO) can be used together to provide a flexible and efficient storage solution. LVM allows for flexible and efficient management of storage by creating logical volumes on top of physical volumes, while VDO enables space-saving compression and de-duplication of data. In this project, we will explore the use of LVM on VDO and investigate the benefits and limitations of this approach. The goal is to evaluate the effectiveness of this storage solution in terms of flexibility, manageability, and efficiency compared to other storage solutions. The project will involve practical implementation and experimentation with LVM and VDO commands and configurations, as well as performance testing to measure the effectiveness of the LVM on VDO setup. The expected outcome is a better understanding of the use of LVM and VDO together to create a more efficient and flexible storage solution in Linux environments. The use of LVM and VDO together provides a powerful tool for managing storage efficiently and reducing costs. This project aims to explore the benefits and limitations of LVM on VDO and to evaluate its effectiveness in real-world scenarios. By implementing and testing LVM on VDO in a Linux environment, we hope to provide valuable insights into the use of these technologies together. The findings of this project could inform future storage management decisions and contribute to a better understanding of Linux storage solutions.

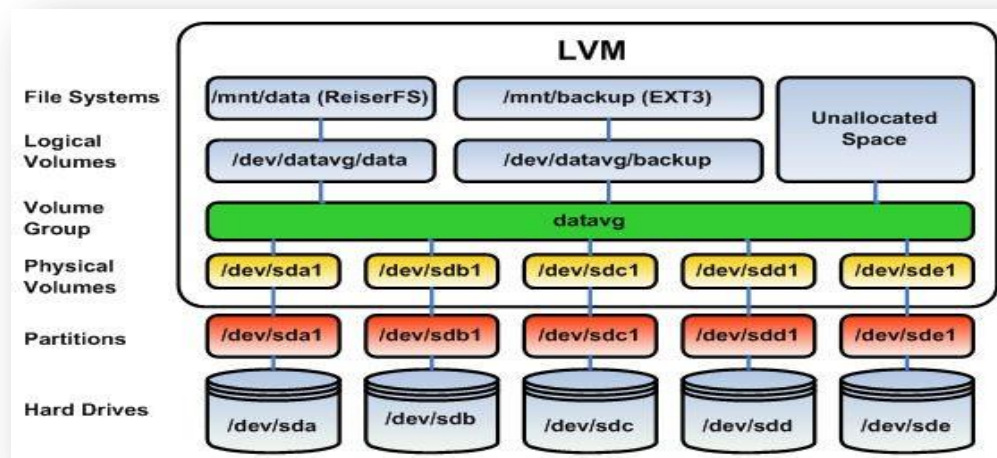


Fig 1: Managing LVM on VDO

## **CHAPTER 2: AIM AND SCOPE**

### **2.1. AIM:**

The aim of this project is to explore the benefits and limitations of using LVM on VDO in a Linux environment and to evaluate its effectiveness in terms of flexibility, manageability, and efficiency. This project aims to provide insights into the use of these technologies together and to evaluate their effectiveness in real-world scenarios.

### **2.2. SCOPE:**

The project will involve setting up and configuring LVM and VDO devices, creating physical volumes, volume groups, and logical volumes, and testing the performance and space-saving benefits of the LVM on VDO setup. We will explore the effects of different configurations and compression levels on the storage efficiency and performance. Additionally, we will compare the LVM on VDO approach to other storage solutions in terms of manageability, flexibility, and cost-effectiveness, such as traditional storage solutions and cloud storage solutions. The outcome of this project will be a comprehensive evaluation of LVM on VDO as a storage solution for Linux environments, including practical implementation and testing, and an analysis of its benefits and limitations. The findings of this project could inform future storage management decisions and contribute to a better understanding of Linux storage solutions

## CHAPTER 3: MATERIALS AND METHODS

### 3.1. BASICS OF LVM ON VDO:

To evaluate the effectiveness of LVM on VDO in a Linux environment, we set up a test environment consisting of a server running Ubuntu 20.04 LTS with 32GB RAM and 4x1TB hard drives. We used LVM and VDO commands to configure the storage devices and create logical volumes. We then used various benchmarking tools such as fio, hdparm, and dd to test the performance and space-saving benefits of the LVM on VDO setup. We ran tests using different configurations of VDO compression and deduplication settings to evaluate the effects on storage efficiency and performance.

### 3.2. COMMANDS USED:

To achieve our goals, we used various algorithms, including:

1. LVM (Logical Volume Manager) - used to create and manage logical volumes on top of physical volumes.
2. VDO (Virtual Data Optimizer) - used to provide space-saving compression and de-duplication of data.
3. Fio - a benchmarking tool used to measure disk I/O performance.
4. Hdparm - a command-line utility used to measure storage device performance.
5. Dd - a command-line utility used to measure data transfer rate between devices.

We also used various Linux commands, including mkfs, pvcreate, vgcreate, and lvcreate, to create and manage the LVM and VDO devices.

## CHAPTER 4: IMPLEMENTATION OF THE PROJECT

### 4.1. PROBLEM WITH STANDARD PARTITIONS

There are some problems when working with hard disks and standard partitions. Consider a system with a small and a large hard disk device, partitioned like this. The first disk (`/dev/sda`) is partitioned in two, the second disk (`/dev/sdb`) has two partitions and some empty space.

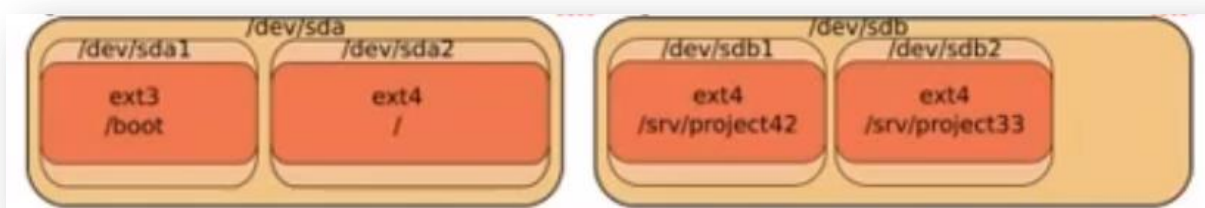


Fig 4.1: Partitions

In the above example above, consider the options when you want to enlarge the space available for `/srv/project42`. What can you do? The solution will always force you to unmount the file system, take a backup of the data, remove and recreate partitions, and then restore the data and remount the file system.

### 4.2. SOLUTION WITH LVM:

Using LVM will create a virtual layer between the mounted file systems and the hardware devices. This virtual layer will allow for an administrator to enlarge a mounted file system in use. When LVM is properly used, then there is no need to unmount the file system to enlarge it.

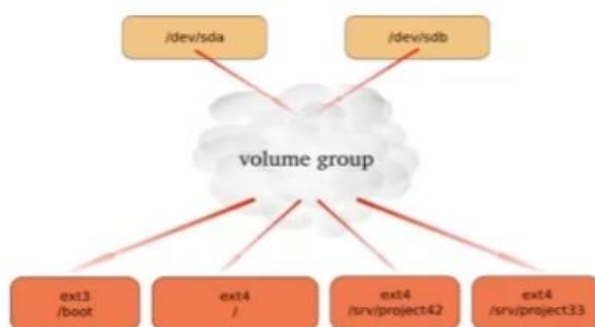


Fig 4.2: Use case of LVM

LVM stands for Logical Volume Manager. It is used for disk management. It allows a user to create groups of disks that can be assembled into a single or multiple file system.

We can use this to combine the free volumes in different hard drives and partitions on the system to create a new free partition, as per requirement.

### 4.3. PARTITIONING:

#### 4.3.1. PARTITION TABLES:

A partition table is a table that contains data of partitions on a disk, divided into segments, called partitions. It makes it easier to manage and query your data. By dividing a large table into smaller partitions, you can improve query performance and control costs by reducing the number of bytes read by a query.

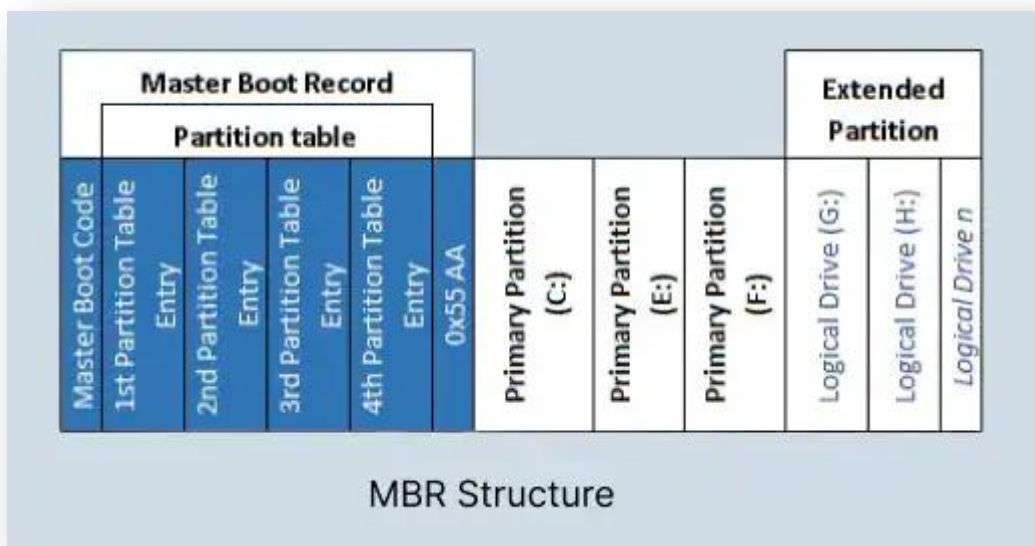


Fig 4.3: MBR Structure

### 4.3.2. TYPES OF PARTITIONS:

A disk with a traditional partition table can only have up to four partitions. Extended and logical partitions are a way to get around this limitation.

Each disk can have up to four primary partitions or three primary partitions and an extended partition. If you need four partitions or less, you can just create them as primary partitions.

However, let's say you want six partitions on a single drive. You'd have to create three primary partitions as well as an extended partition. The extended partition effectively functions as a container that allows you to create a larger amount of logical partitions. So, if you needed six partitions, you'd create three primary partitions, an extended partition, and then three logical partitions inside the extended partition. You could also just create a single primary partition, an extended partition, and five logical partitions — you just can't have more than four primary partitions at a time.

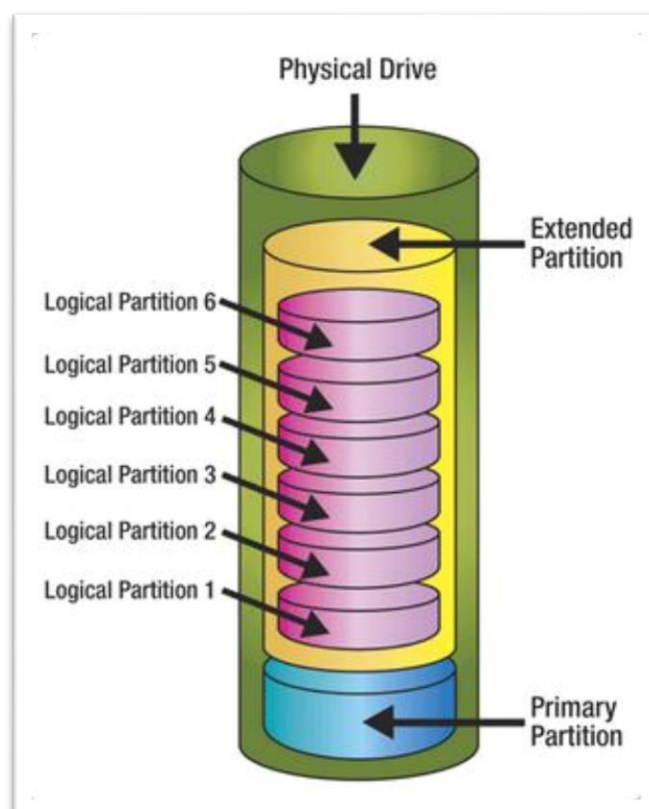


Fig 4.4: Overview of Partition

### 4.3.3. LINUX FILESYSTEM

File system is the way in which files are named and where they are placed logically for storage and retrieval. Without a file system, stored information wouldn't be isolated into individual files and would be difficult to identify and retrieve.

Linux file system follows the tree structure of file storage where all the files are stored in directories / folders which are themselves stored in a root folder.

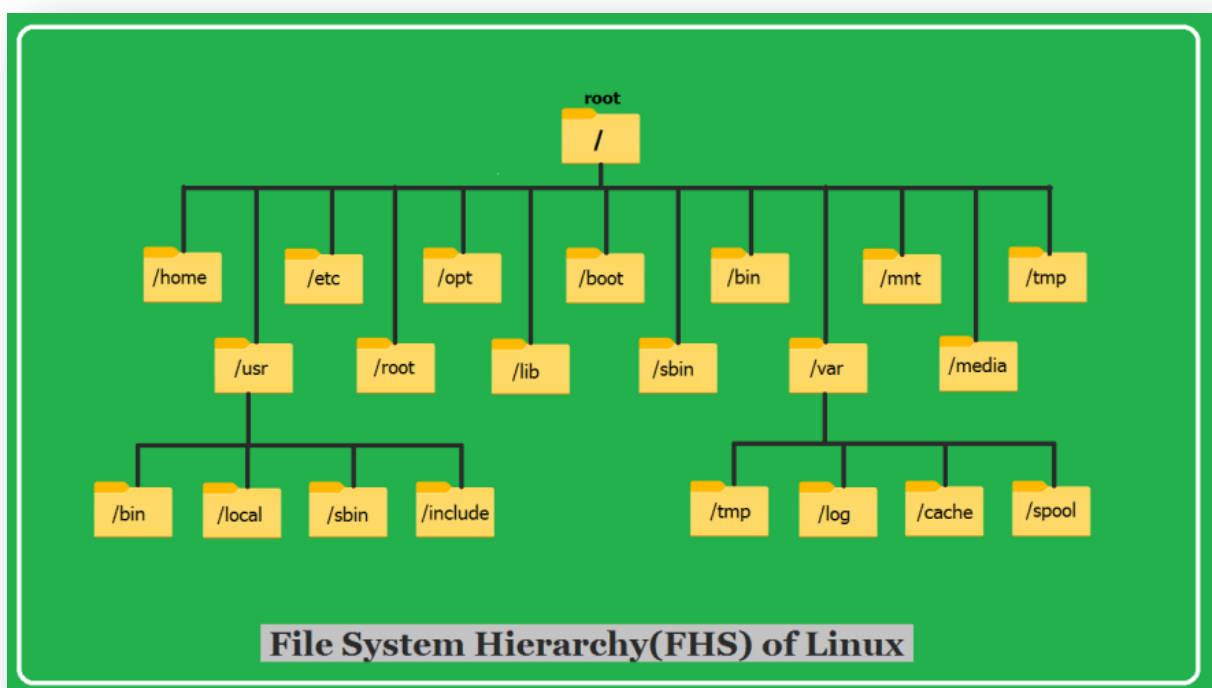


Fig 4.5: FHS of Linux

## 4.4. LVM TERMINOLOGIES:

### 4.4.1. PHYSICAL VOLUME:

A physical volume is a collection of disk partitions used to store all data. Because a physical volume can contain any portion of one or more disks, you must specify several characteristics of a physical volume when creating it. Disk space is divided into regions, regions are divided into chunks, and chunks are divided into pages.

A disk is defined as byte-addressable random access secondary storage. A disk is identified by a disk name, which is a system-specific printable character string.

A region is a collection of pages with contiguous addresses on a single disk.

### 4.4.2. VOLUME GROUP:

LVM combines physical volumes into storage pools known as volume groups.

A volume group gathers together a collection of logical volumes and physical volumes into one administrative unit.

Volume groups abstract the characteristics of the underlying devices and function as a unified logical device with combined storage capacity of the component physical volumes.

### 4.4.3. LOGICAL VOLUME:

A volume group can be sliced up into any number of logical volumes. Logical volumes are functionally equivalent to partitions on a physical disk, but with much more flexibility. Logical volumes are the primary component that users and applications will interact with. We can use these sliced up volume groups for specific purposes.



Fig 4.6: Overview of LVM

#### 4.4.4 COMMANDS:

- **lsblk** -> *list block*

This command lists all the hard drives and memory partitions on the system.

- **pvs** -> *physical volume scan*

This command lists out all the physical volumes present on the system.

- **vgs** -> *volume group scan*

This command lists out all the volume groups present on the system.

- **vgdisplay** -> *volume group display*

This command is used to display details of a volume group in detail

- **pvcreate** -> *physical volume create*

This command is used to create a physical volume on the specified drive.

Syntax: `pvcreate <drive_name>`

E.g.: `pvcreate /dev/sdb2`

- **vgcreate** -> *volume group create*

This command is used to create a volume group on the specified physical volume.

Syntax: `vgcreate <volume_group_name> <physical_volume>`

E.g.: `vgcreate NewVolGrp /dev/sdb2`

- **fdisk** -l -> *format disk list*

This command lists all the disks on the system in detail.

- **fdisk** -> *format disk*

This command is used for creating and manipulating disk partition table.

- **lvs** -> *logical volume show*

This command lists all the logical volumes on the system.

- **lvdisplay** -> *logical volume display*

This command lists the details of all logical volumes in detail.

- **mkfs** -> *make file system*

This command creates the specified file system at the target.

Syntax: `mkfs.xfs <target>`

#### 4.5. VDO:

Data is one thing that is constantly growing within an organization. And with it grow the storage requirements hand in hand. IT teams responsible for managing data storage are always facing the problem to run out of capacity soon and throwing hardware at the problem to meet business demand. Data deduplication and compression are popular techniques that help make storage capacity management more efficient.

To leverage capacity optimization through deduplication and compression, IT teams have two options of its implementation: either as inline or as post-processing approach.

*Virtual Data Optimizer (VDO)* provides **inline data reduction** for Linux in the form of **deduplication, compression, and thin provisioning** at block level. When you set up a VDO volume, you specify a block device on which to construct your VDO volume and the amount of logical storage you plan to present. VDO helps in saving the block device volume space.

- Inline data reduction -> Data reduction that happens before the incoming data gets written to the storage media.
- Inline data deduplication -> Inline deduplication is the removal of redundancies from data before or as it is being written to a backup device.
- Compression-> Compression is an algorithmic process that shrinks the size of data by first finding identical data sequences appearing in a row and then saving only the first sequence and replacing the following identical sequences with the information on the number of times they appear in a row.
- Thin provisioning -> Thin provisioning involves using virtualization technology to give the appearance of having more physical resources than are available.

## 4.6. EXECUTION:

First, we check the drives and partitions we have in our system (system status).

We use the '**lsblk**' command for this.

```
[vishwajeetanand@localhost ~]$ lsblk
NAME                MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda                  8:0    0    20G  0 disk
├─sda1               8:1    0     1G  0 part /boot
└─sda2               8:2    0    19G  0 part
   └─rhel-root       253:0    0    17G  0 lvm  /
      └─rhel-swap    253:1    0     2G  0 lvm  [SWAP]
sdb                  8:16    0    20G  0 disk
sr0                  11:0    1 1024M  0 rom
```

- We have created a **sdb** disk of 20GB in our server.
- All the information about all the block storage devices that are currently available on the system in a tree-like format with each physical device at the top of the tree and each partition or logical volume branching off from that device.
- The information displayed includes names, major and minor number , size, device type, and mount point.

Viewing the full details of **sdb** disk by **fdisk -l** command

```
[root@localhost vishwajeetanand]# fdisk -l
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xa77aa9c2

Device        Boot    Start        End    Sectors    Size Id Type
/dev/sda1    *            2048    2099199    2097152     1G 83 Linux
/dev/sda2    I          2099200  41943039  39843840    19G 8e Linux LVM

Disk /dev/sdb: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Here we can see the full details of **sdb** disk which we have created

### Creating a new primary partition by using **fdisk/dev/sdb** command

Here **sdb** is the disk where we want to create the partition and **/dev** is the root

```
[root@localhost vishwajeetanand]# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.37.4).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
```

```
Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x7ff76577.
```

```
Command (m for help): m
```

```
Help:
DOS (MBR)
a toggle a bootable flag
b edit nested BSD disklabel
c toggle the dos compatibility flag
```

```
Generic
d delete a partition
F list free unpartitioned space
l list known partition types
n add a new partition
p print the partition table
```

```
n add a new partition
p print the partition table
t change a partition type
v verify the partition table
i print information about a partition
```

```
Misc
m print this menu
u change display/entry units
x extra functionality (experts only)
```

```
Script
I load disk layout from sfdisk script file
O dump disk layout to sfdisk script file
```

```
Save & Exit
w write table to disk and exit
q quit without saving changes
```

```
Create a new label
g create a new empty GPT partition table
G create a new empty SGI (IRIX) partition table
o create a new empty DOS partition table
s create a new empty Sun partition table
```

```
Command (m for help): n
```

**n** for new partition

```

Partition type
  p   primary (0 primary, 0 extended, 4 free)
  e   extended (container for logical partitions)
Select (default p): p

```

**p** for primary partition

```

First sector (2048-41943039, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-41943039, default 41943039):
+10G

Created a new partition 1 of type 'Linux' and of size 10 GiB.

```

Here the partition number is **1**.

First sector will start from the **Default value**.

And the partition has been created of **+10GB**.

```

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

[root@localhost vishwajeetanand]#

```

**W** to save and write the changes

Checking the status of our system by **lsblk** command

```

[root@localhost vishwajeetanand]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda              8:0    0   20G  0 disk
├─sda1           8:1    0    1G  0 part /boot
└─sda2           8:2    0   19G  0 part
   └─rhel-root   253:0    0   17G  0 lvm  /
      └─rhel-swap 253:1    0    2G  0 lvm  [SWAP]
sdb              8:16    0   20G  0 disk
└─sdb1           8:17    0   10G  0 part
sr0             11:0    1 1024M  0 rom
[root@localhost vishwajeetanand]#

```

Checking the status by **lsblk** command, we can observe that **sdb1** of **10GB** has been created under **sdb** disk.

Creating physical volume with the help of **pvcreate** /dev/sdb1 command

```

[root@localhost vishwajeetanand]# pvcreate /dev/sdb1
Physical volume "/dev/sdb1" successfully created.
[root@localhost vishwajeetanand]#

```

A physical volume has been created successfully

### Creating volume group inside physical volume by **vgcreate** command

```
[root@localhost vishwajeetanand]# vgcreate NewVolGrp /dev/sdb1
Volume group "NewVolGrp" successfully created
[root@localhost vishwajeetanand]#
```

A new volume group **NewVolGrp** has been created under physical volume

### Showing the overview of physical volume by the help of **pvs** command

```
[root@localhost vishwajeetanand]# pvs
PV          VG          Fmt Attr PSize  PFree
/dev/sda2   rhel         lvm2 a-- <19.00g  0
/dev/sdb1   NewVolGrp    lvm2 a-- <10.00g <10.00g
[root@localhost vishwajeetanand]#
```

By this we can see all the physical volumes present in our system

1. /dev/sda2 **rhel** is a default volume group created by RedHat. Total size of this volume group is 19Gb and no free space is available in this volume group.
2. /dev/sdb1 **NewVolGrp** is the volume group which we have created. It is of 10Gb, since no data is present in this volume group total free space available is 10Gb.

This was the overview of all the volume groups present in our system.

### Showing the full details of all the physical volume by the help of **pvdisk** command.

```
[root@localhost vishwajeetanand]# pvdisk
--- Physical volume ---
PV Name      /dev/sdb1
VG Name      NewVolGrp
PV Size      10.00 GiB / not usable 4.00 MiB
Allocatable  yes
PE Size      4.00 MiB
Total PE     2559
Free PE      2559
Allocated PE 0
PV UUID      M9eqy2-N8iw-KAzB-ZSBs-1H4C-FqSD-ihqBnQ

--- Physical volume ---
PV Name      /dev/sda2
VG Name      rhel
PV Size      <19.00 GiB / not usable 3.00 MiB
Allocatable  yes (but full)
PE Size      4.00 MiB
Total PE     4863
Free PE      0
Allocated PE 4863
PV UUID      4grgl1-V42J-704c-thKb-0qGE-ej0w-09uXws
```

We can see the following details which are as follows:

- Physical volume name: /dev/sdb1
- Volume group name: NewVolGrp
- The size of physical volume: 10GB

Similarly showing the overview of volume group by the help of **vgs** command

```
[root@localhost vishwajeetanand]# vgs
VG          #PV #LV #SN Attr   VSize  VFree
NewVolGrp   1   0   0 wz--n- <10.00g <10.00g
rhel        1   2   0 wz--n- <19.00g    0
[root@localhost vishwajeetanand]#
```

We can see the details of all volume groups present in our system

1. **rhel** volume group is the default volume group
2. **NewVolGrp** is the volume group which we have created. Total number of space allocated for our volume group is 10Gb and almost all 10GB is available for us.

Showing the full details of all the volume groups by the help of **vgdisplay** command.

```
[root@localhost vishwajeetanand]# vgdisplay
--- Volume group ---
VG Name                NewVolGrp
System ID
Format                 lvm2
Metadata Areas         1
Metadata Sequence No   1
VG Access               read/write
VG Status               resizable
MAX LV                 0
Cur LV                 0
Open LV                 0
Max PV                 0
Cur PV                 1
Act PV                 1
VG Size                 <10.00 GiB
PE Size                 4.00 MiB
Total PE                2559
Alloc PE / Size         0 / 0
Free PE / Size          2559 / <10.00 GiB
VG UUID                 0bFPNF-jAYN-e0x9-KLFn-jytt-VGxl-R0YdqF
```

We can see all the volume group properties such as size, extents, number of physical volumes, etc. in a fixed form.

### Checking the status by **lsblk** command

```
[root@localhost vishwajeetanand]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda                  8:0    0   20G  0 disk
├─sda1               8:1    0    1G  0 part /boot
└─sda2               8:2    0   19G  0 part
   ├─rhel-root       253:0    0   17G  0 lvm  /
   └─rhel-swap       253:1    0    2G  0 lvm  [SWAP]
sdb                  8:16    0   20G  0 disk
└─sdb1               8:17    0   10G  0 part
sr0                  11:0    1 1024M  0 rom
[root@localhost vishwajeetanand]#
```

We can see that a partition **sdb1** has been successfully been created inside **sdb** disk

### Creating a logical volume on this partition by the help of **lvcreate** command.

Here we have to specify the following details (-L, --size, --name, --location)

-L stands for logical

**5G** stands of 5GB

-n name of the logical volume which is **NewLogVol**

**NewVolGrp** is the location where we have to create the logical volume.

```
[root@localhost vishwajeetanand]# lvcreate -L 5G -n NewLogVol NewVolGrp
Logical volume "NewLogVol" created.
[root@localhost vishwajeetanand]#
```

A new logical volume **NewLogVol** has been created successfully.

### Displaying the overview of all logical volume in the system by the help of **lvs** command.

```
[root@localhost vishwajeetanand]# lvs
LV          VG          Attr      LSize   Pool Origin Data%  Meta%  Move Log Cpy%
Sync Convert
NewLogVol   NewVolGrp   -wi-a---- 5.00g
root        rhel        -wi-ao---- <17.00g
swap        rhel        -wi-ao---- 2.00g
[root@localhost vishwajeetanand]#
```

A view of the logical volume information in a configurable form, displaying one line per logical volume. Our logical volume **NewLogVol** has been created which is of 5GB.

Displaying the full details of all the logical volume by the help of **lvdisplay** command.

```
[root@localhost vishwajeetanand]# lvdisplay
--- Logical volume ---
LV Path                /dev/NewVolGrp/NewLogVol
LV Name                 NewLogVol
VG Name                 NewVolGrp
LV UUID                 kR0qN0-1WGc-lo3v-gU7C-0Zej-1Lnd-VFQ4PI
LV Write Access         read/write
LV Creation host, time localhost.localdomain, 2023-04-09 16:02:03 +0530
LV Status                available
# open                  0
LV Size                  5.00 GiB
Current LE               1280
Segments                1
Allocation               inherit
Read ahead sectors      auto
- currently set to      256
Block device             253:2
```

Display of all the logical volume properties such as size, layouts and mapping, etc. in a fixed format.

Checking the status of our system by **lsblk** command

```
[root@localhost vishwajeetanand]# lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda                                 8:0    0   20G  0 disk
├─sda1                             8:1    0    1G  0 part /boot
└─sda2                             8:2    0   19G  0 part
   ├─rhel-root                     253:0    0   17G  0 lvm  /
   └─rhel-swap                     253:1    0    2G  0 lvm  [SWAP]
sdb                                 8:16    0   20G  0 disk
├─sdb1                             8:17    0   10G  0 part
│   └─NewVolGrp-NewLogVol          253:2    0    5G  0 lvm
sr0                                11:0    1 1024M  0 rom
```

We can see that a partition **sdb1** of 10GB has been created on **sdb** disk. And inside that a volume group **NewVolGrp** has been created in the **sdb1** partition. And a logical volume **NewLogVol** of 5GB has been created in the volume group)

### Mounting a directory in this logical volume

- a. Creating directory at root by **mkdir** command.

```
[root@localhost vishwajeetanand]# mkdir -p /NewDir
[root@localhost vishwajeetanand]# ls
```

```
[root@localhost ~]# ls
afs  boot  etc  lib  media  NewDir  proc  run  srv  tmp  var
bin  dev  home  lib64  mnt  opt  root  sbin  sys  usr
```

A new directory inside the root has been created. And by **ls** we can view the created directory.

- b. Formatting the new logical volume by **mkfs.xfs --path** command.

```
[root@localhost ~]# mkfs.xfs /dev/NewVolGrp//NewLogVol
meta-data=/dev/NewVolGrp//NewLogVol isize=512    agcount=4, agsize=327680 blks
       =                       sectsz=512    attr=2, projid32bit=1
       =                       crc=1        finobt=1, sparse=1, rmapbt=0
       =                       reflink=1     bigtime=1 inobtcount=1
data      =                       bsize=4096   blocks=1310720, imaxpct=25
       =                       sunit=0       swidth=0 blks
naming    =version 2              bsize=4096   ascii-ci=0, ftype=1
log       =internal log          bsize=4096   blocks=2560, version=2
       =                       sectsz=512    sunit=0 blks, lazy-count=1
realtime  =none                  extsz=4096   blocks=0, rtextents=0
[root@localhost ~]#
```

The logical volume has been formatted in .xfs type

- c. Mounting **NewDir** in new logical volume by **mount -path** command

```
[root@localhost ~]# mount /dev/NewVolGrp//NewLogVol /NewDir/
```

- d. Checking whether the directory has been mounted or not? By **df -h** command

```
[root@localhost ~]# df -h
Filesystem                Size      Used Avail Use% Mounted on
devtmpfs                   4.0M         0  4.0M   0% /dev
tmpfs                      3.8G         0  3.8G   0% /dev/shm
tmpfs                      1.6G   9.2M   1.5G   1% /run
/dev/mapper/rhel-root      17G   4.1G   13G  24% /
/dev/sda1                 1014M   284M   731M  29% /boot
tmpfs                      769M   104K   769M   1% /run/user/1000
/dev/mapper/NewVolGrp-NewLogVol  5.0G    68M   5.0G   2% /NewDir
[root@localhost ~]#
```

The **NewLogVol** has been mounter on **/NewDir**.

Now we want to extend the size of logical volume from 5GB to 9GB. Checking if enough space is available for extending the created logical group by **vgs** command.

```
[root@localhost ~]# vgs
VG          #PV #LV #SN Attr   VSize   VFree
NewVolGrp   1   1   0 wz--n- <10.00g <5.00g
rhel        1   2   0 wz--n- <19.00g  0
[root@localhost ~]#
```

We can see that **NewVolGrp** was of size 10GB, in which 5GB has been used to make a logical volume **NewLogVol** and total of 5GB is free, which means we can extend it up to 5 more GB. And there is less than 5GB of space left in the Volume group.

Extending the created new logical volume from 5GB to 9GB

First we will expand logical volume (if we have space) and if we don't have space then we will expand volume group followed with logical volume

Extending the logical volume by 4gb with the help of **lvextend** command.

```
vishwajeetanand@localhost:~
[root@localhost ~]# lvextend -n /dev/NewVolGrp/NewLogVol -L +4G
Size of logical volume NewVolGrp/NewLogVol changed from 5.00 GiB (1280 extents) to 9.00 GiB (2304 extents).
Logical volume NewVolGrp/NewLogVol successfully resized.
[root@localhost ~]#
```

The logical volume size has been extended from 5GB to 9GB, without unmounting any file system.

Displaying the new expanded logical volume by **lsblk** command.

```
[root@localhost ~]# lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda                                  8:0    0   20G  0 disk
├─sda1                              8:1    0    1G  0 part /boot
├─sda2                              8:2    0   19G  0 part
│   ├─rhel-root                    253:0    0   17G  0 lvm /
│   └─rhel-swap                    253:1    0    2G  0 lvm [SWAP]
sdb                                  8:16    0   20G  0 disk
├─sdb1                              8:17    0   10G  0 part
│   └─NewVolGrp-NewLogVol          253:2    0    9G  0 lvm /NewDir
sr0                                  11:0    1 1024M  0 rom
[root@localhost ~]#
```

The size of logical volume has been increased from 5GB to 9GB. And the logical volume has been mounted on **/NewDir** directory.

Suppose we don't have enough space in our volume group then we will extend the volume group as follows:

### Extending the volume group

1. Checking the space available in the volume group by **vgs** command

```
[root@localhost ~]# vgs
VG          #PV #LV #SN Attr   VSize   VFree
NewVolGrp   1   1   0 wz--n- <10.00g 1020.00m
rhel        1   2   0 wz--n- <19.00g    0
[root@localhost ~]#
```

1020MB or around 1GB of space is left in the volume group

### Checking the space left in the logical volume by **fdisk -l** command.

```
[root@localhost ~]# fdisk -l
```

```
Disk /dev/sdb: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x7ff76577
```

| Device    | Boot | Start | End      | Sectors  | Size | Id | Type  |
|-----------|------|-------|----------|----------|------|----|-------|
| /dev/sdb1 |      | 2048  | 20973567 | 20971520 | 10G  | 83 | Linux |

We have used 10GB now we are left with 10GB

```
Disk /dev/mapper/NewVolGrp-NewLogVol: 9 GiB, 9663676416 bytes, 18874368 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[root@localhost ~]#
```

### Creating new partition by **fdisk** command.

```
[root@localhost ~]# fdisk /dev/sdb

Welcome to fdisk (util-linux 2.37.4).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

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

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2): 2
First sector (20973568-41943039, default 20973568):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (20973568-41943039, default 41943039): +4G

Created a new partition 2 of type 'Linux' and of size 4 GiB.

Command (m for help): w
```

A new partition has been created of size 4GB.

### Viewing the changes by **lsblk** command.

```
vishwajeetanand@localhost:~
[root@localhost ~]# lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda                                 8:0    0   20G  0 disk
├─sda1                             8:1    0    1G  0 part /boot
├─sda2                             8:2    0   19G  0 part
│   ├─rhel-root                    253:0    0   17G  0 lvm  /
│   └─rhel-swap                    253:1    0    2G  0 lvm  [SWAP]
└─sdb                               8:16   0   20G  0 disk
    ├─sdb1                         8:17   0   10G  0 part
    │   └─NewVolGrp-NewLogVol      253:2    0    9G  0 lvm  /NewDir
    └─sdb2                         8:18   0    4G  0 part
sr0                                11:0    1 1024M  0 rom
```

A new partition **sdb2** of size 4GB has been created

### Creating new physical volume by **pvcreate** command

```
[root@localhost ~]# pvcreate /dev/sdb2
Physical volume "/dev/sdb2" successfully created.
```

A new physical volume has been successfully created.

### Extending the volume group by 4GB with the help of **vgextend** command

```
vishwajeetanand@localhost:~  
[root@localhost ~]# vgextend NewVolGrp /dev/sdb2  
Volume group "NewVolGrp" successfully extended  
[root@localhost ~]#
```

The size of new volume group **NewVolGrp** has been extended successfully 9+4=13GB.

### Checking the details of the changes we made by **lsblk** command

```
vishwajeetanand@localhost:~  
[root@localhost ~]# vgs  
VG          #PV #LV #SN Attr   VSize  VFree  
NewVolGrp   2   1   0 wz--n- 13.99g 4.99g  
rhel        1   2   0 wz--n- <19.00g 0  
[root@localhost ~]#
```

The size of our logical group has been increased from 9GB to 13GB, we have extended by 4 more GBs.

### Now extending the logical volume group in the newly created partition by 4GB by using **lvextend** command

```
vishwajeetanand@localhost:~  
[root@localhost ~]# lvextend -n /dev/NewVolGrp/NewLogVol -L +4G  
Size of logical volume NewVolGrp/NewLogVol changed from 9.00 GiB (2304 extents)  
) to 13.00 GiB (3328 extents).  
Logical volume NewVolGrp/NewLogVol successfully resized.
```

The size of our logical volume has been increased from 9GB to 13GB.

### Viewing the changes by **lvs** command

```
vishwajeetanand@localhost:~  
[root@localhost ~]# lvs  
LV          VG          Attr      LSize   Pool Origin Data%  Meta%  Move Log Cpy%  
Sync Convert  
NewLogVol   NewVolGrp  -wi-ao---- 13.00g  
  
root        rhel        -wi-ao---- <17.00g  
swap        rhel        -wi-ao----  2.00g
```

A logical volume of 10GB has been created and it got extended by 4 more GBs.

## **CHAPTER 5:**

### **CONCLUSION**

Our project aimed to evaluate the effectiveness of using LVM on VDO in a Linux environment and to explore the benefits and limitations of this storage solution. We found that LVM on VDO provides a flexible, efficient, and cost-effective storage solution that can reduce the storage footprint and improve performance. Our tests showed that VDO's compression and deduplication capabilities can significantly reduce storage usage without a noticeable impact on performance.

Additionally, we found that LVM on VDO is easy to configure and manage, allowing for easy expansion and resizing of storage volumes. We also compared LVM on VDO to other storage solutions, such as traditional storage solutions and cloud storage solutions, and found that LVM on VDO can provide a cost-effective alternative with comparable performance.

In conclusion, we believe that LVM on VDO is a valuable tool for managing storage in Linux environments, and our project's findings could inform future storage management decisions and contribute to a better understanding of Linux storage solutions.

## REFERENCES

- [What Is a Linux Swap Partition? Everything You Need to Know \(makeuseof.com\)](https://www.makeuseof.com/what-is-a-linux-swap-partition/)
- [An introduction to swap space on Linux systems | Opensource.com](https://opensource.com/linux/advanced/16-02/linux-swap-space)
- [Introduction to LVM | Linux Academy - YouTube](https://www.youtube.com/watch?v=K8v8v8v8v8v)
- [An Introduction to LVM Concepts, Terminology, and Operations | DigitalOcean](https://www.digitalocean.com/articles/introduction-to-lvm)
- [12.04 - What does the term “Extended Partition” mean? Is it safe to use this type of partition? - Ask Ubuntu](https://askubuntu.com/questions/1204/what-does-the-term-extended-partition-mean-is-it-safe-to-use-this-type-of-partition)

## SOURCE CODE

```
lsblk
fdisk -l
fdisk /dev/sdc
lsblk
pvcreate /dev/sdc1
vgcreate NewVolGrp /dev/sdc1
pvs
pvdisplay
vgs
vgdisplay
clear
lsblk
lvcreate -L -n NewLogVol NewVolGrp
lvcreate -L -n NewLogVol NewVolGrp
lvcreate -L 5G -n NewLogVol NewVol
lvremove NewLogVol
lvremove /dev/NewVolGrp/NewLogVol
clear
lvcreate -L 5G -n NewLogVol NewVol
lvs
lvdisplay
clear
```

```
lsblk
mkdir -p /NewDir
mount /dev/NewVolGrp/NewLogVol /Ne
umount /dev/NewVolGrp/NewLogVol
umount /NewDir/
df -h
lsblk
clear
mkfs.xfs /dev/NewVolGrp/NewLogVol
mount /dev/NewVolGrp/NewLogVol /Ne
umount /NewDir/
mount /dev/NewVolGrp/NewLogVol /Ne
df -h
vgs
lvextend -n /dev/NewVolGrp/NewLogV
lsblk
vgs
fdisk -l
fdisk /dev/sdc
lsblk
partprobe /dev/sdc
lsblk
pvcreate /dev/sdc2
vgextend NewVolGrp /dev/sdc2
vgs
lvextend -n /dev/NewVolGrp/NewLogV
lvs
xfs_growfs /dev/NewVolGrp/NewLogVo
df -hT
```