

# **Project: Resizing FileSystem Safely (ext4 and xfs), Repair Corrupt FileSystem**

**Doneby: Aashish Chaudhary**

---

**Concepts and Tools:** parted, partprobe, resize2fs, xfsdump, xfsrestore, e2fsck, fsck, losetup, and others.

## **Overview:**

---

In this lab, I have demonstrated how to resize filesystem in Linux System safely. I have used **ext4** and **xfs** filesystem for this purpose. While doing this lab, I have encountered multiple tools, issues and troubleshooted them. Each section has all the steps in detail, and along with this documentation, I have also added CLI command documentation, that has all the necessary steps and commands required for this project.

---

## **What happens in the project?**

I will keep everything short and concise what I did in this project. I have added the following picture to understand what happens in the project.

### **a. First step:**

I created a rawdisk from / filesystem and mounted it to the mountPoint1.

Step1:

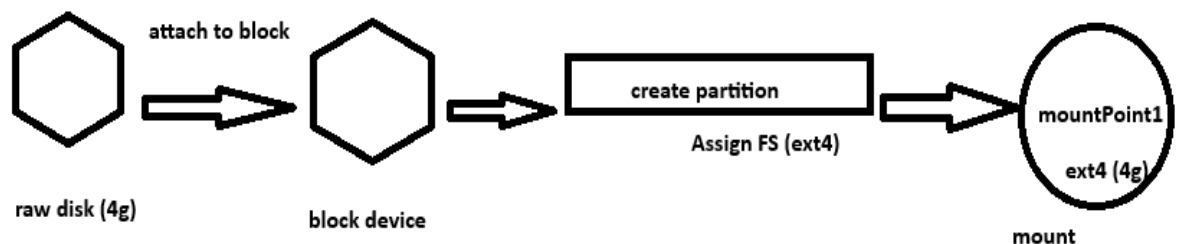
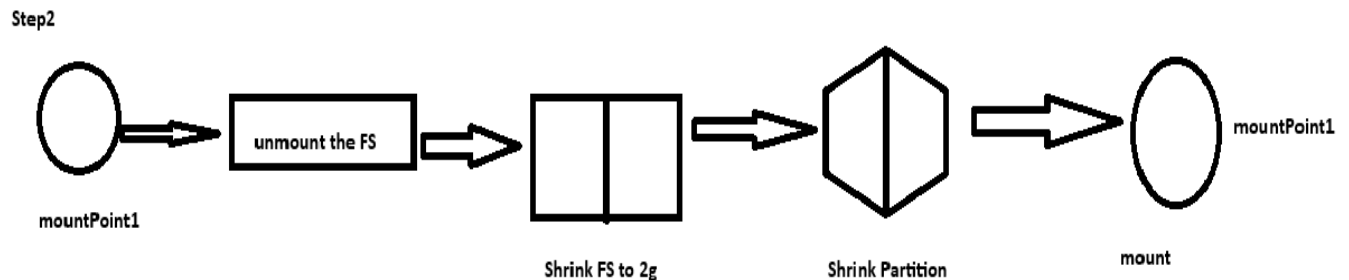


Fig. mount point1

**b. Second step:**

I shrunk the filesystem from 4g to 2g and also resized the partition to have 2g. and mounted the filesystem to the same mountPoint1.

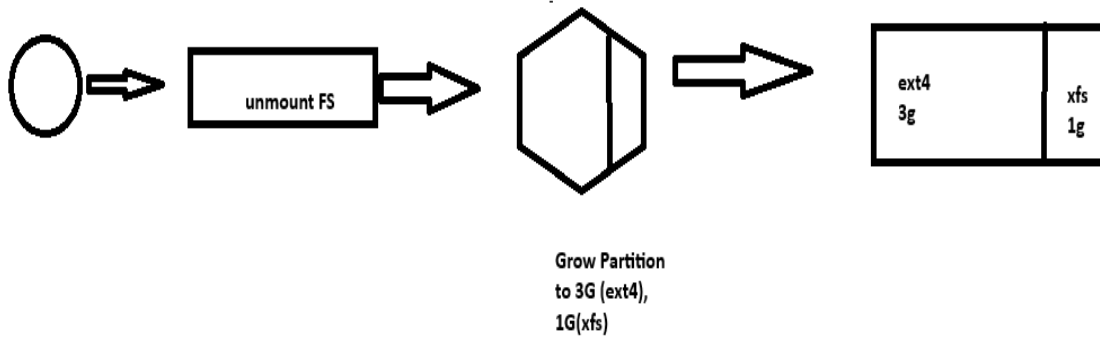


**c. Third step:**

Then, I grew the filesystem from 2g to 3g. This was done exactly opposite to what we did in the shrinking step. We should reverse the steps of shrinking Filesystem and Partition, while we try to grow the size. Otherwise, the filesystem would try to write data on so called “**not yet available**” place, because the underlying partition doesn’t have enough blocks needed for the filesystem to write data.

In addition, to the size grow, I also added one more partition for **xfs** filesystem for remaining 1g and created **xfs filesystem** on that partition. Later, I also resized that partition and filesystem as well.

Step 3:



#### **d. Fourth Step:**

This step involves two mini steps:

1. First, shrink the ext4 from 3G to 2G, so that we can grow the size of the partition that holds xfs fs.
2. Second, the actual xfs growth itself.

In the first step, I followed the same steps what I did on second steps. It's same; we unmount, resize the Filesystem, partition then mount it. It's pretty simple. However, in second section, growing xfs filesystem is different than ext4 filesystem. The general rule is the filesystem should be mounted, and once the partition table is increased, we can run **sudo xfs growfs /dev/loopop2** to adjust the filesystem size to the partition. It is easier and unidirectional, meaning when we try to move the endpoint far, it happens easily. But, what if, we have more space before the starting point of the partition.

My case was the latter one. I had to find alternatives, and this is why I chose to backup mountpoint2 to the separate drive, delete existing partition and wiping off the xfs filesystem and later on restore what I had backed up. The problem was that my machine was running out of space.

To solve this, I added actual storage in my Proxmox and create a new logical volume and backed data into it using xfsdump tool. I have shown how I created a new drive below:

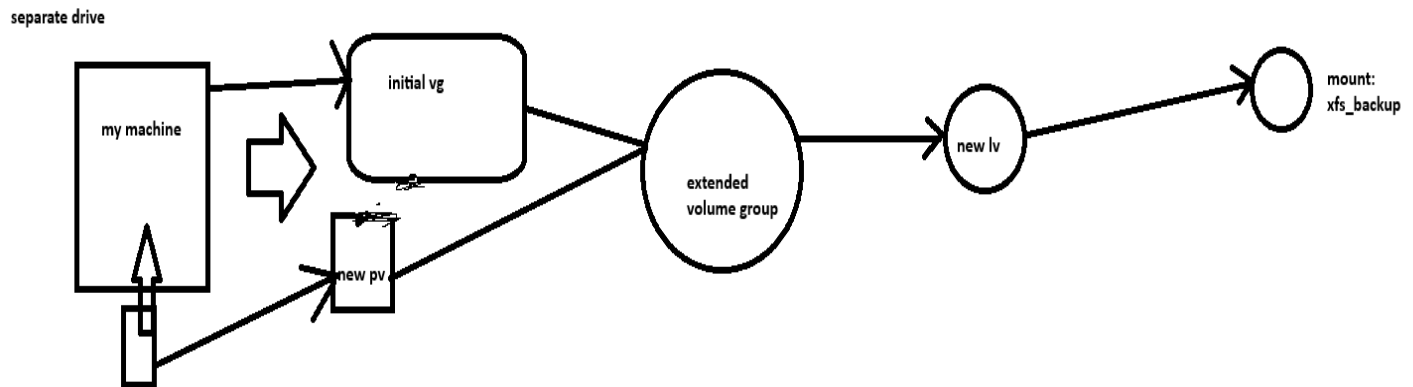


Fig: xfs\_backup created

After this, the following steps were carried out:

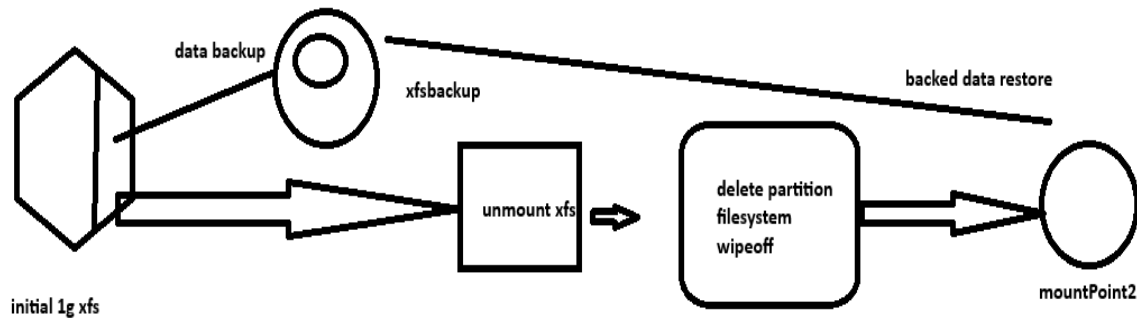


Fig. xfs grow and backup restore.

### Issues and Troubleshooting:

**Corrupt FileSystem:** In my second step, during shrinking filesystem from 4g to 2g, I resized the filesystem first and then resized the partition table to 2g. After the

partition table update, and running e2fsck to check any inconsistencies, the filesystem got corrupted. The problem was superblock information mismatch. The filesystems' superblock thought there were more blocks available where as physical device itself had less blocks.

### **How I Solved this?**

First, I had to understand the problem what it was, I read the error message, did not understand what it meant. After researching for a while, I understood, the filesystem thinks it had more blocks than actually what was available. I tried many steps: running e2fsck, fsck -f, fsck with previous backup. Later, simple observation solved the issue. I thought, why don't I just try to resize the filesystem itself to have the same number of blocks as the physical device already had. That simple question took care of it.

In short, I have demonstrated how to resize the filesystem safely in the Linux environment. The resizing steps differ from one filesystem to another. I have initially defined a whole 4g disk to be one partition and ext4 filesystem. Later I have done both growing and shrinking, and while doing so, in the remaining space, I have initially defined xfs system and then grown its space, as well. The CLI document also documents all the steps and commands clearly. If followed, it is much easier to do the lab, and review them later.