LVM in Linux — Create and Extend a Logical Volume

A guide on how to create and extend an LVM logical volume on an XFS filesystem

Yama Hakimi · Follow
10 min read · Mar 28, 2022

7

Photo by Benjamin Lehman on Unsplash

Let’s get started by talking a bit about LVM

Logical Volume Manager deals with the storage in a way that is, by far, more efficient than traditional disk management. With standard disk partitioning, the storage capacity is based on the individual disk capacity, but with LVM, the storage space is managed by combining all the available physical hard drives as if they are part of a pool, making them usable as a whole, instead of handling them individually.

Let’s say we have four 1TB drives, with a traditional disk scheme you would handle them individually but with LVM, these four 1TB drives would be considered to be this one 4TB single chunk or aggregated storage capacity. This gives us greater flexibility and control over the disk layout and allows us to manipulate disks in an easier way. One of the main benefits of using LVM is the ability to effortlessly grow the filesystem.

To understand and use LVM we need to comprehend three main components, they are interconnected and together make what we call a *Logical Volume,* these components are:

- Physical Volume
- Volume Group
- Logical Volume

Physical Volume: These are the base block used to create an LVM, physical volumes or "PVs" are simply your **physical storage devices**, whether it is SSD or HDD drives. For a hard drive to be considered a physical volume, it has to be initialized marked as a physical volume so it can be used by the LVM.

Volume Group: We can think of a volume group "VG" as a pool that is comprised of physical volumes. Let's say we three 1TB SSD hard drives that are part of a volume group, this "VG" will show up as having a consolidated storage capacity of 3TB, which will then be used to create logical volumes.

Logical Volumes: When our VG is created, we can finally create a logical volume. We can carve one or more logical volumes from a single volume group. A logical volume will be treated as a traditional partition that would be then mounted on a directory and be in use.

The figure below illustrates the structure of a Logical Volume:

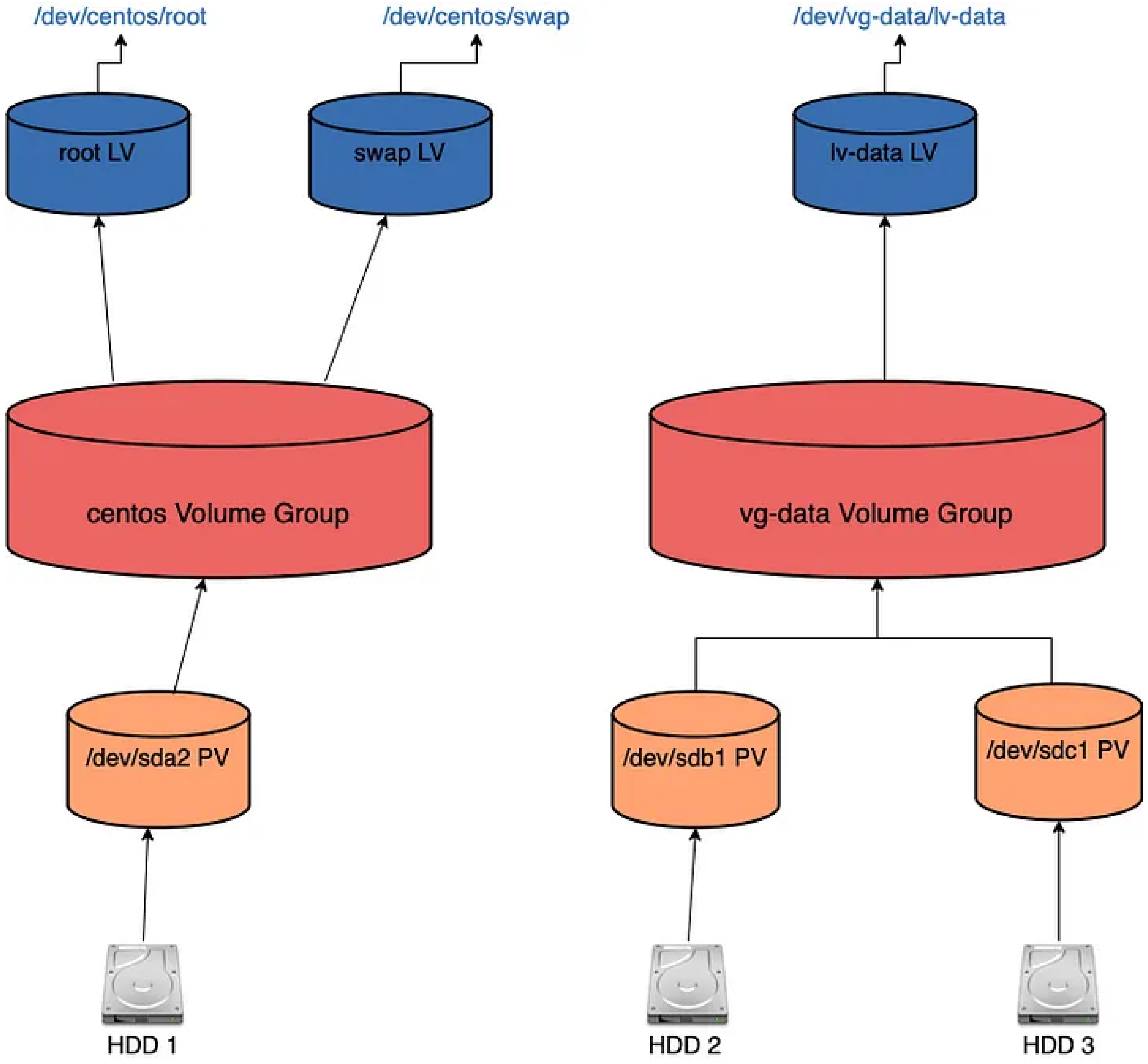


Figure 1: LVM structure

(In this particular example, we happen to have multiple Logical Volumes and Volume groups as it is how the server we use for the demo is configured.)

It starts from physical devices, the hard drives, that are used to create our Physical Volumes or PVs. In this example, we have three separate HDDs, each is used to create one Physical Volume. The partition /dev/sda2 will make our first PV, /dev/sdb1 the second and sdc1 the third.

Moving up we have two separate Volume groups and each of these have their distinctive Logical Volumes, as we said earlier, LVs are carved out from Volume groups, we may have one or many Logical Volumes coming from one VG.

The final layer in this abstraction is the Logical Volume, for example, lv-data that comes from the vg-data volume group, once lv-data is ready it can be formatted and used as a mount point, in this case: **/dev/vg-data/lv-data**.

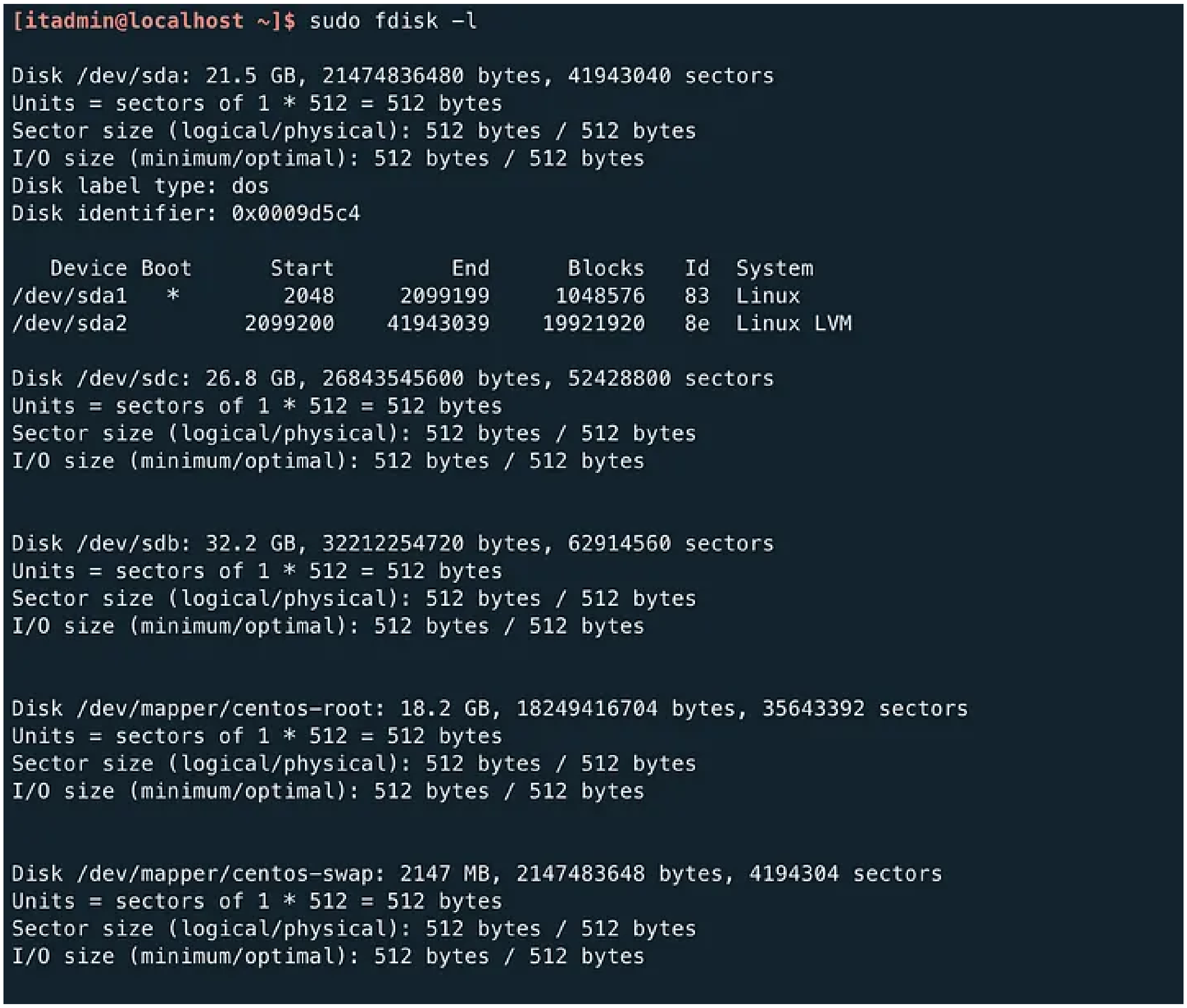
How to create an LVM Logical Volume:

Before diving into extending/growing an existing Logical Volume, let's see how we can create one from the scratch on a system where it has not been set up before, and then, we shall extend it in the next section. If you only wish to extend an existing LV, you can skip this section.

To set up a new Logical Volume, we need to proceed in the following order:

1. Create physical volume or volumes from the existing hard drives.
2. Create a Volume group and add the physical volumes to it.
3. Create a Logical Volume from the Volume Group.
4. Format the Logical Volume as required — xfs, ext4 etc.
5. Finally, mount the new filesystem.

— We do need to have disk space available to create a physical volume. On this server (a virtual machine) we have two separate raw hard drives that we will be using for our Logical Volume creation demo.

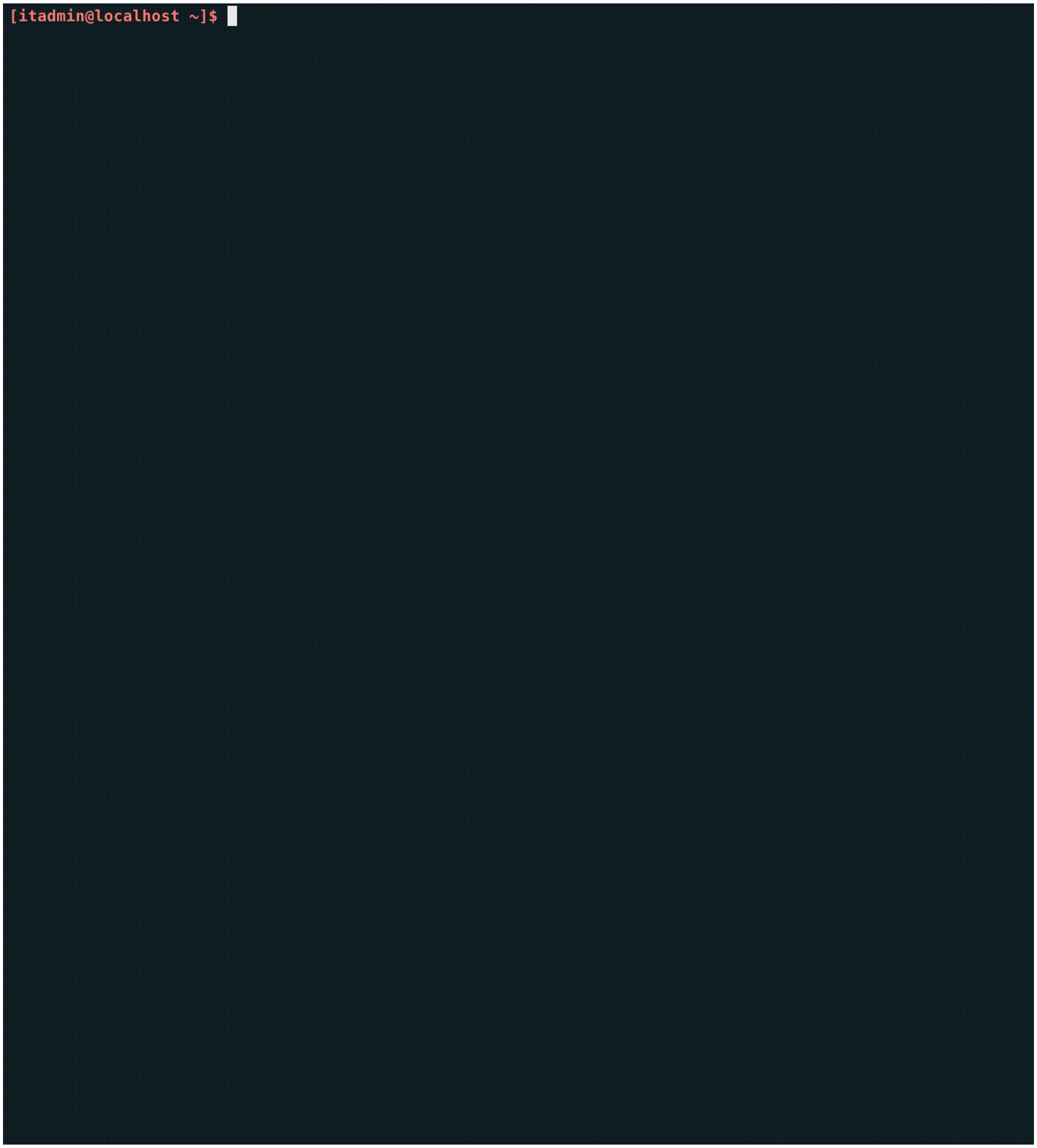


```
[itadmin@localhost ~]$ sudo fdisk -l

Disk /dev/sda: 21.5 GB, 21474836480 bytes, 41943040 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
Disk label type: dos
Disk identifier: 0x0009d5c4

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200    41943039    19921920   8e  Linux LVM

Disk /dev/sdc: 26.8 GB, 26843545600 bytes, 52428800 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


Disk /dev/sdb: 32.2 GB, 32212254720 bytes, 62914560 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


Disk /dev/mapper/centos-root: 18.2 GB, 18249416704 bytes, 35643392 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


Disk /dev/mapper/centos-swap: 2147 MB, 2147483648 bytes, 4194304 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
```

Output of fdisk — listing the partitions present on the server.

We can see we have **/dev/sdb** and **/dev/sdc** which are two raw disks, they are not yet available and cannot be used, so we need to format them.

Let's look at this quick demo on how we can make this new disk /dev/sdb available:



```
[itadmin@localhost ~]$
```

Creating a new partition on /dev/sdb

A few things about what we have done:

- We are making this raw new drive **/dev/sdb** available by creating a new partition and assigning the proper label to it.

- We have not specified a specific size for this partition but chose the default which is to use the whole disk, hitting enter without specifying any size will allocate the whole space available.
- The crucial part here is to choose "**8e**" Hex code which will label this new partition and assign it as an LVM type, if we wanted to create an ext4 partition then the label code would be different. For every new partition created, it has to be labelled before it can be used.
- "w" is required to save the changes we have made.

Now that our disk is ready, let's start by creating a physical volume with **pvcreate**:

```
[itadmin@localhost ~]$ sudo pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created.
[itadmin@localhost ~]$
```

Our Physical volume has been created! Let's check whether our PV was indeed created with the help of **pvdisplay** (pvs can also be used):

```
[itadmin@localhost ~]$ sudo pvdisplay
[sudo] password for itadmin:
  --- Physical volume ---
  PV Name               /dev/sda2
  VG Name               centos
  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               sa3xFR-jbDb-SKuS-yQN1-VzzW-bveG-aA0yha

 "/dev/sdb1" is a new physical volume of "<30.00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb1
  VG Name
  PV Size               <30.00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               Q54fqC-Np3X-XUaF-j5vE-Px29-pT9H-JYsDUh

[itadmin@localhost ~]$
```

We can see we already had a PV /dev/sda2 meaning the root partition is also built on top of a logical volume. The new disk /dev/sdb1 is now a physical volume. We can notice that the VG Name of this new physical volume is empty! that is because it has not been added to a volume group yet!
Let’s do that right now:

```
[itadmin@localhost ~]$ sudo vgcreate vg-data /dev/sdb1
  Volume group "vg-data" successfully created
[itadmin@localhost ~]$
```

- we use **vgcreate** to create a new volume group, assigning a name to this new volume group and adding the physical volume or volumes to it, in this case, we make /dev/sdb1 part of this VG.

Let’s check our newly created volume group:

```
[itadmin@localhost ~]$ sudo vgs
  VG      #PV #LV #SN Attr   VSize   VFree
  centos    1   2   0 wz--n- <19.00g      0
  vg-data   1   0   0 wz--n- <30.00g <30.00g
[itadmin@localhost ~]$
```

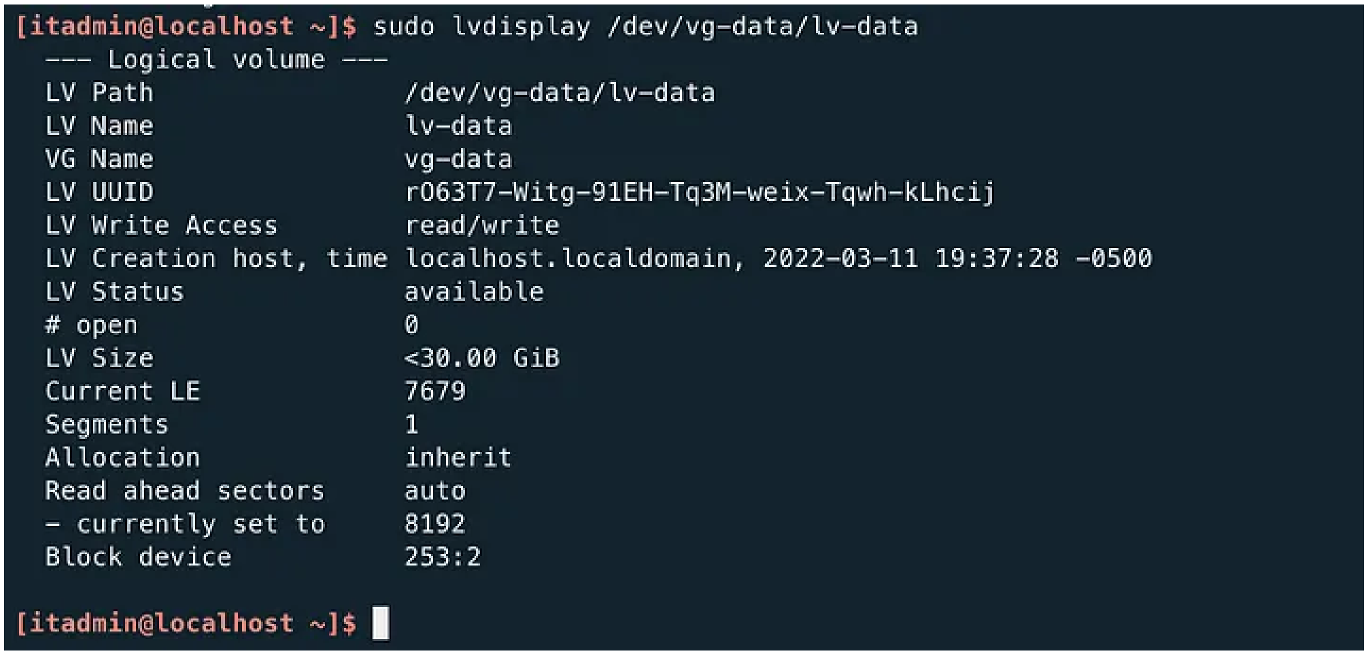
- vg-data is indeed there and has 30GB free — which is the size of the physical volume(the new partition) we added earlier.

Now, we create the logical volume with **lvcreate:**

```
[itadmin@localhost ~]$ sudo lvcreate --name lv-data -l 100%FREE vg-
data
 Logical volume "lv-data" created.
[itadmin@localhost ~]$
```

- -l : is used to specify how much space we wish to take from the volume group, here we allocate 100% of it, we do need to mention the volume group name in our command.

Checking the new Logical Volume with **lvdisplay** :



```
[itadmin@localhost ~]$ sudo lvdisplay /dev/vg-data/lv-data
  --- Logical volume ---
  LV Path                /dev/vg-data/lv-data
  LV Name                lv-data
  VG Name                vg-data
  LV UUID                rO63T7-Witg-91EH-Tq3M-weix-Tqwh-kLhcij
  LV Write Access        read/write
  LV Creation host, time localhost.localdomain, 2022-03-11 19:37:28 -0500
  LV Status              available
  # open                 0
  LV Size                <30.00 GiB
  Current LE             7679
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:2

[itadmin@localhost ~]$
```

The output of lvdisplay on lv-data

Our Logical Volume has been created!

The final step we should take to be able to use this logical volume is to format this new LV, we do this with the help of the **mkfs.xfs** command:

```
[itadmin@localhost ~]$ sudo mkfs.xfs /dev/vg-data/lv-data
[sudo] password for itadmin:
meta-data=/dev/vg-data/lv-data   isize=512    agcount=4,
agsize=1965824 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=0, sparse=0
data     =                       bsize=4096   blocks=7863296,
imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=3839, version=2
         =                       sectsz=512   sunit=0 blks, lazy-
count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
[itadmin@localhost ~]$
```

. **mkfs** stands for "make a filesystem", which is what it does. In this case, we want to have an xfs filesystem hence mkfs.xfs. The xfs filesystem is an upgrade from ext4 in many aspects and is the default filesystem with RHEL servers, that said, there are advantages of using ext4 over xfs, depending on the situation.

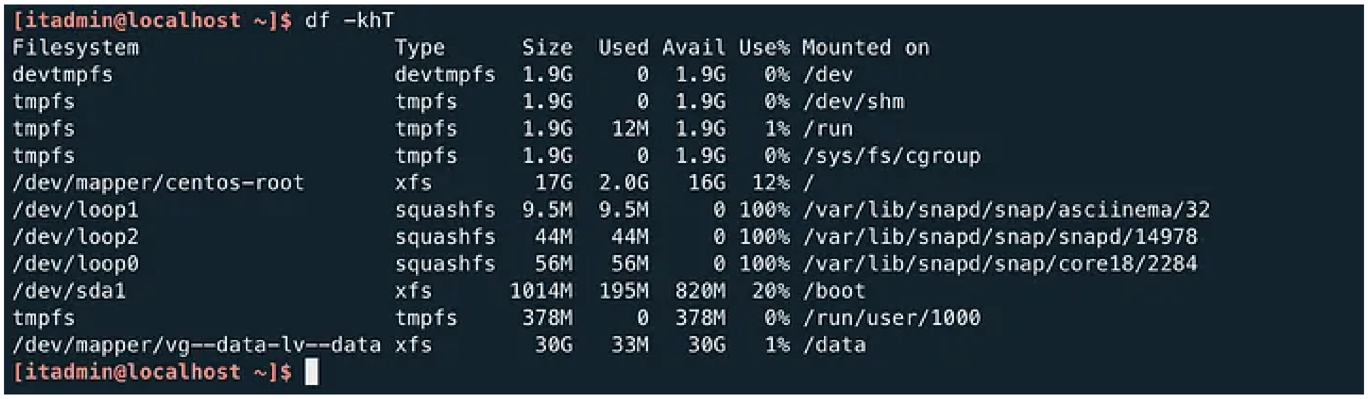
We want to mount this newly created logical volume on a mount point, let's create it and proceed:

```
[itadmin@localhost ~]$ sudo mkdir /data
[itadmin@localhost ~]$ sudo mount /dev/vg-data/lv-data /data
```

We created a brand new directory where our new filesystem will be mounted, anything under /data will belong to this new logical volume.

— As a side note — we do need to add any new mount point in the: **/etc/fstab** file so it persists throughout reboots.

Let's have a look at our newly mounted **/data** directory with the df command:



```
[itadmin@localhost ~]$ df -khT
Filesystem                    Type      Size  Used Avail Use% Mounted on
devtmpfs                      devtmpfs  1.9G     0  1.9G   0% /dev
tmpfs                         tmpfs     1.9G     0  1.9G   0% /dev/shm
tmpfs                         tmpfs     1.9G   12M  1.9G   1% /run
tmpfs                         tmpfs     1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mapper/centos-root       xfs        17G  2.0G   16G  12% /
/dev/loop1                    squashfs  9.5M  9.5M     0 100% /var/lib/snapd/snap/asciinema/32
/dev/loop2                    squashfs   44M   44M     0 100% /var/lib/snapd/snap/snapd/14978
/dev/loop0                    squashfs   56M   56M     0 100% /var/lib/snapd/snap/core18/2284
/dev/sda1                     xfs      1014M  195M  820M  20% /boot
tmpfs                         tmpfs     378M     0  378M   0% /run/user/1000
/dev/mapper/vg--data-lv--data xfs        30G   33M   30G   1% /data
[itadmin@localhost ~]$
```

Output of the df -khT command

- We need to check whether the filesystem is indeed XFS hence the "T" switch in our df command, which is used to show the filesystem type.
- Our logical volume lv-data is all ready and mounted on /data!

Task completed!

How to extend a Logical Volume

We have seen how to create a logical volume from scratch, but in most cases, you will need to increase the size of an already existing logical volume so it can accommodate more data.

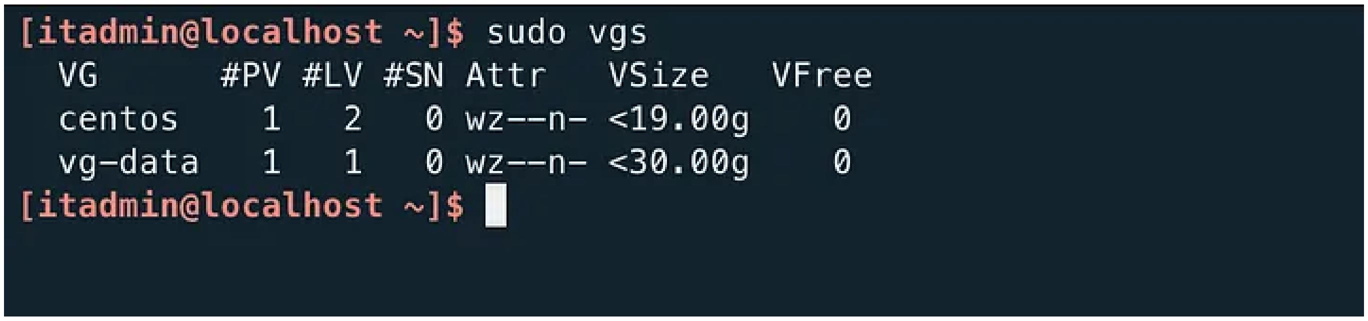
Let's say you have plugged in additional hard drives into your server and now need to make them part of an existing Logical Volume (grow an existing LV) Let's jump right in and do this:

First and foremost, to extend an existing Logical Volume, we **must have free space in the Volume Group.**

This is the sequence we need to follow to extend a Logical Volume:

> Add new hard drives -> create new physical volumes -> Add the new PVs to the exiting Volume Group -> Extend the Logical Volume.

First of all, let's display some info about the current state of our Volume group, vg-data is the volume group we are concerned about:

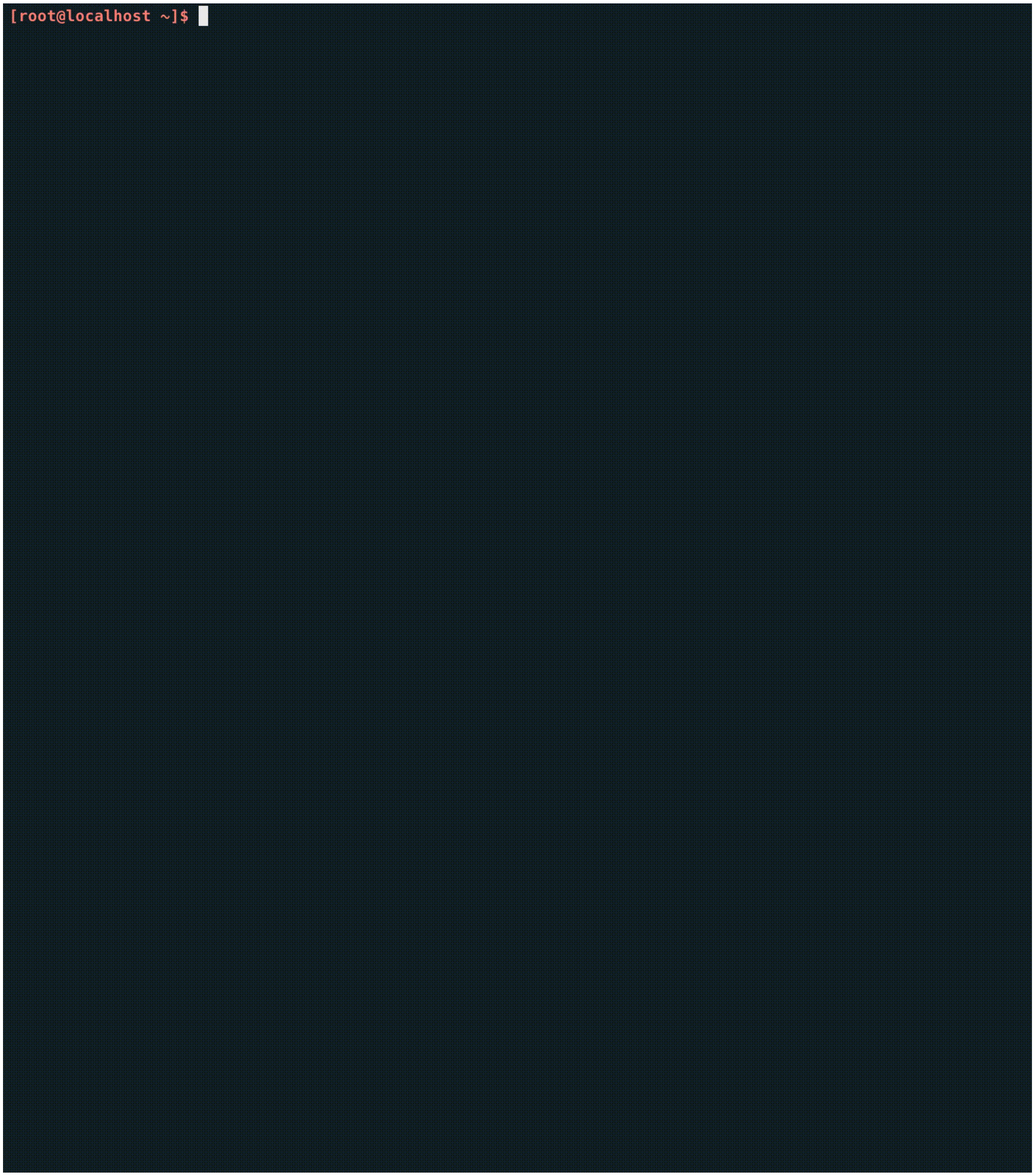


```
[itadmin@localhost ~]$ sudo vgs
  VG      #PV #LV #SN Attr   VSize   VFree
  centos    1   2   0 wz--n- <19.00g    0
  vg-data   1   1   0 wz--n- <30.00g    0
[itadmin@localhost ~]$
```

Checking the vg-data Volume Group

- We can see that the Volume Group **vg-data** has no free space available, let's add a new physical volume to it.

First, we need to make this new drive usable by creating a new partition and assigning the "LVM" label to it, we do this using fdisk, as below:



```
[root@localhost ~]$ 
```

Making /dev/sdc available for the new physical volume.

- Using fdisk, we created a new partition, assigned the whole size by not specifying any size (when pressing enter without any value, it assigns whatever is available)

- We need to label it properly, **8e** is the label needed to make it the "LVM type".

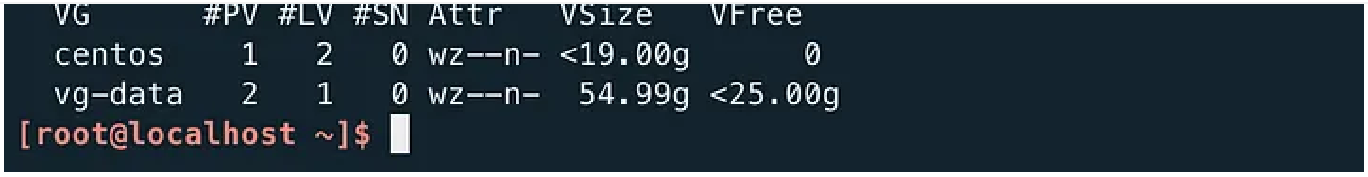
Our disk is ready, and we can now create the physical volume:

```
[root@localhost ~]$ pvcreate /dev/sdc1
  Physical volume "/dev/sdc1" successfully created.
[root@localhost ~]$
```

Created! Now, let's add it to the vg-data Volume Group, we do this with the help of the **vgextend** command:

```
[root@localhost ~]$ vgextend vg-data /dev/sdc1
  Volume group "vg-data" successfully extended
[root@localhost ~]$
```

Our VG has been extended! Let's double-check:

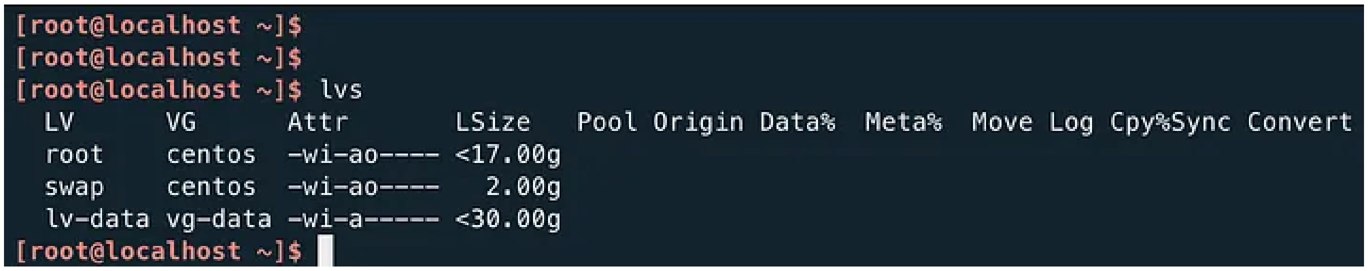


```
  VG       #PV #LV #SN Attr   VSize   VFree
  centos     1   2   0 wz--n- <19.00g      0
  vg-data    2   1   0 wz--n-  54.99g <25.00g
[root@localhost ~]$
```

The Volume Group has now free space.

Perfect! we can see that the vg-data has now 25GB of free space.

Let's jump on our Logical Volume now, right now, this is the status of it:



```
[root@localhost ~]$
[root@localhost ~]$
[root@localhost ~]$ lvs
  LV      VG      Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  root    centos  -wi-ao---- <17.00g
  swap    centos  -wi-ao----   2.00g
  lv-data vg-data -wi-a----- <30.00g
[root@localhost ~]$
```

lvs output before extending it.

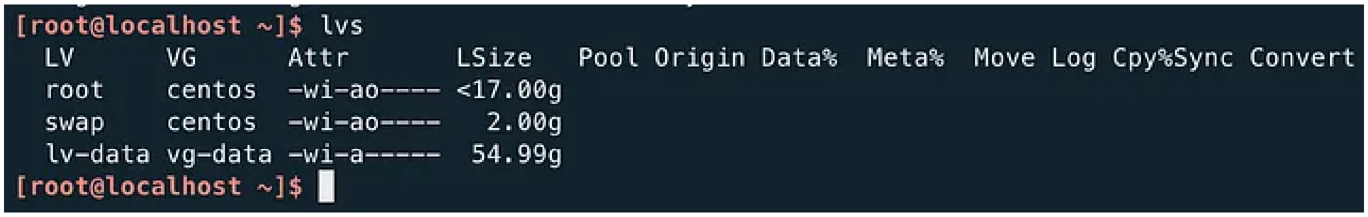
We extend the lv-data Logical Volume with the lvextend command:

```
[root@localhost ~]$ lvextend -l +100%FREE /dev/vg-data/lv-data
  Size of logical volume vg-data/lv-data changed from <30.00 GiB
(7679 extents) to 54.99 GiB (14078 extents).
  Logical volume vg-data/lv-data successfully resized.
[root@localhost ~]$
```

- lvextend will extend the **lv-data** logical volume, the **+100%FREE** option means that the volume will be extended to all the remaining sizes available from the Volume Group.
- If we had wanted to extend the logical volume by a certain size, let's say 5GB, we would have done as such:

```
[root@localhost ~]$ lvextend -L +5G /dev/vg-data/lv-data
```

Now, let's check the status of our Logical Volume:



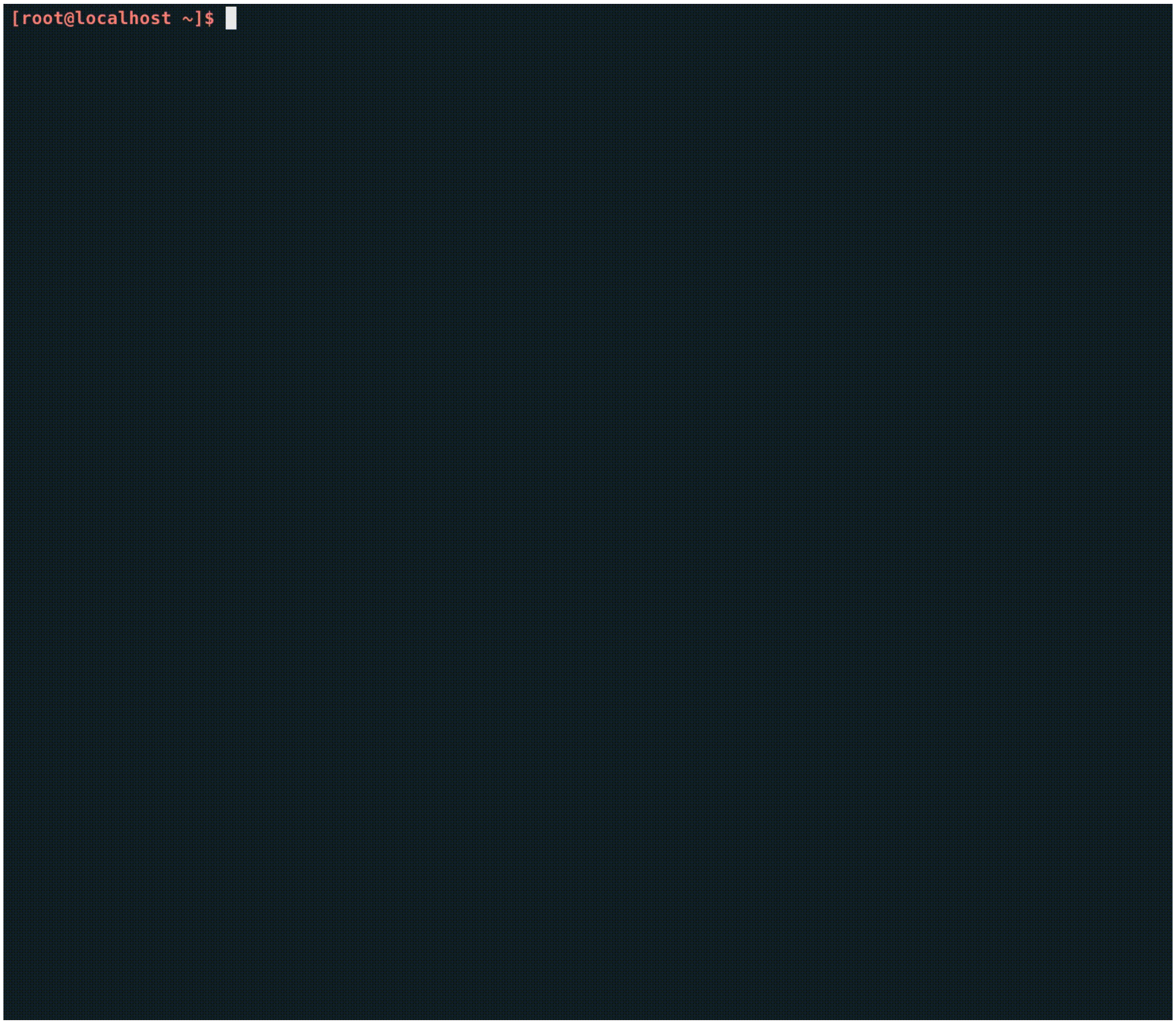
```
[root@localhost ~]$ lvs
  LV      VG      Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  root    centos  -wi-ao---- <17.00g
  swap    centos  -wi-ao----   2.00g
  lv-data vg-data -wi-a-----  54.99g
[root@localhost ~]$
```

Output of lvs after extending the LV.

As per the output, the LSize has indeed increased, the Logical Volume has been extended!

We are not done yet! , one last step is to resize the filesystem so the newly added storage capacity is available, we do that with the **xfs_growfs** command:

```
[root@localhost ~]$ xfs_growfs /dev/vg-data/lv-data
```

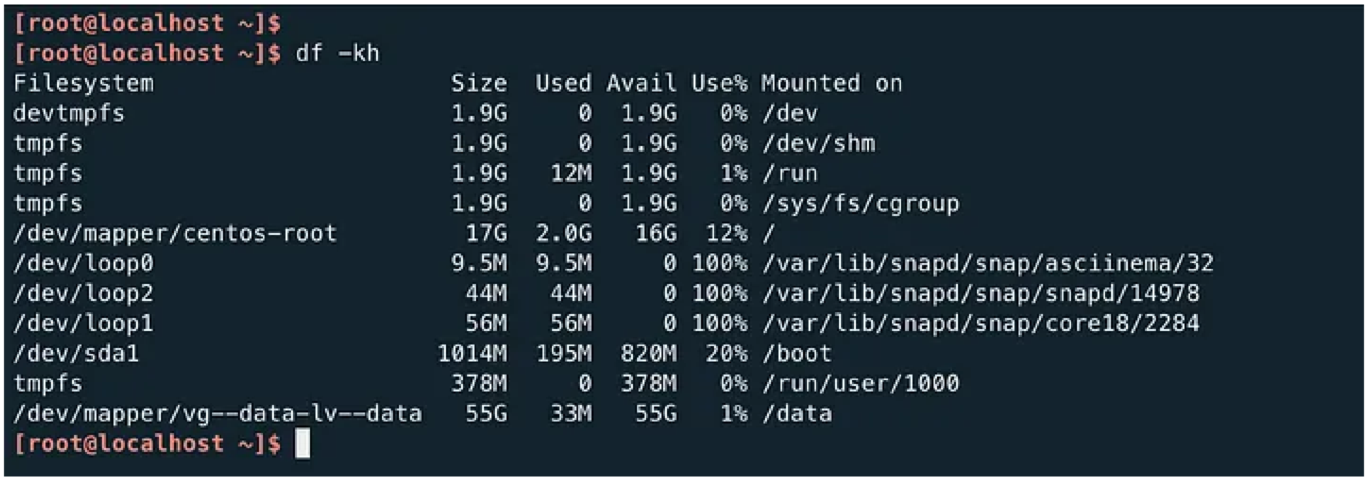


```
[root@localhost ~]$ 
```

Resizing the filesystem.

- we can see that the data blocks have been changed, the filesystem has been extended.
- The other great thing we should mention here is we did not even need to umount our mount point /data when growing the filesystem!

The output of df -kh confirms that the **/data** mount point, which is where the lv-data logical volume is mounted, has been extended:



```
[root@localhost ~]$
[root@localhost ~]$ df -kh
Filesystem                    Size  Used Avail Use% Mounted on
devtmpfs                      1.9G     0  1.9G   0% /dev
tmpfs                         1.9G     0  1.9G   0% /dev/shm
tmpfs                         1.9G   12M  1.9G   1% /run
tmpfs                         1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mapper/centos-root        17G  2.0G   16G  12% /
/dev/loop0                    9.5M  9.5M     0 100% /var/lib/snapd/snap/asciinema/32
/dev/loop2                     44M   44M     0 100% /var/lib/snapd/snap/snapd/14978
/dev/loop1                     56M   56M     0 100% /var/lib/snapd/snap/core18/2284
/dev/sda1                    1014M  195M  820M  20% /boot
tmpfs                         378M     0  378M   0% /run/user/1000
/dev/mapper/vg--data-lv--data  55G   33M   55G   1% /data
[root@localhost ~]$
```

Output of the df -kh

All done, our Logical Volume has been successfully extended!

This was a little about Logical Volume Manager in Linux, how to create and extend a Logical Volume, please feel free to add a comment, suggestion or any input you may have.

Thank you for reading! and see you in the next post :)

Linux Linux Tutorial Lvm Redhat Linux System Administration