Linux Storage

Disk Partitioning

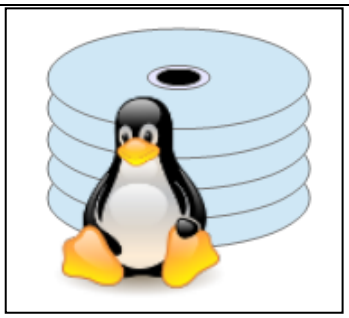


Table of Contents

1. Hard Drive Naming Convention

1.1. References to Hard Drive

/dev/sda = Hard drive 0

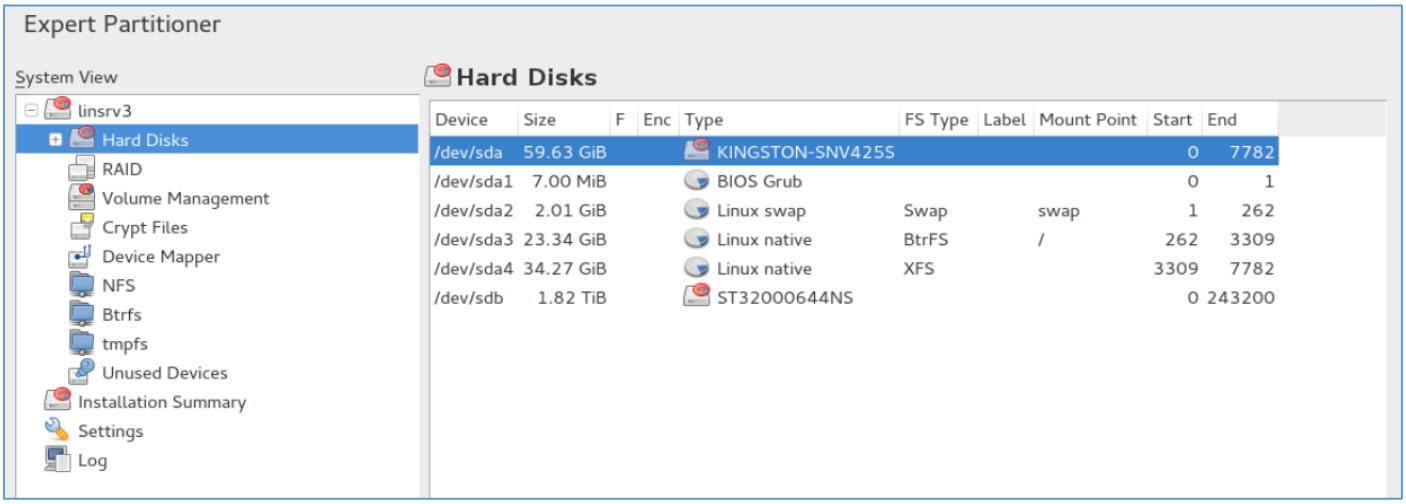
KINGSTON-SNV4255
/dev/sda 59.63 GiB

/dev/sdb = Hard drive 1

SEAGATE Constellation ST32000644NS
ES 2 TB 7200 RPM SATA 3GB/s 64 MB Cache
3.5-Inch Hard Drive

/dev/sdb 1.82 TiB ST32000644NS

/dev/sda /dev/sdb /dev/sdc
dev = device sd = SCSI mass-storage driver



2. Partitions Tables

A partition table describes the layout of partitions of a Hard drive. There are two partition table standards – MBR (Master Boot Record) and GPT (GUID Partition Table). MBR, also know as ms-dos, is the first standard.

MBR's major limitations led to the development of GPT. Those limitations are :

1. It does not allow the configuration of more than four main partitions. Those partitions are called primary partitions.
2. Disk partitions are limited to 2TB

Newer computers comme with a replacement firmware for the old BIOS system called UEFI (Unified Extensible Firmware interface), and GPT is a part of the UEFI standard.

2.1. Partitions and Partition Numbering

To install an operating system on a hard drive, it must first be subdivided into distinct storage units. Those storage units are called **partitions**. Under the MBR partitioning scheme, there are three different types of partitions – Primary, Extended, and Logical. Extended, and Logical partitions will be discussed further down.

With MBR, any partition that is not explicitly created as an extended or logical partition, is a primary partition. And, as stated earlier, there can be no more than four primary partitions. Figure 6 was taken from a Linux installation with four primary partitions. If you observe closely, you will see that the first primary partition is sda1 and the last sda4. Unlike hard drives, partition numbers start from 1, not 0 (zero). Any disk space that's not allocated to the primary partitions is listed as Free or free space. But while it may be free, it is, however, unusable. And that is because as far as the system is concerned, that free space does not exist.

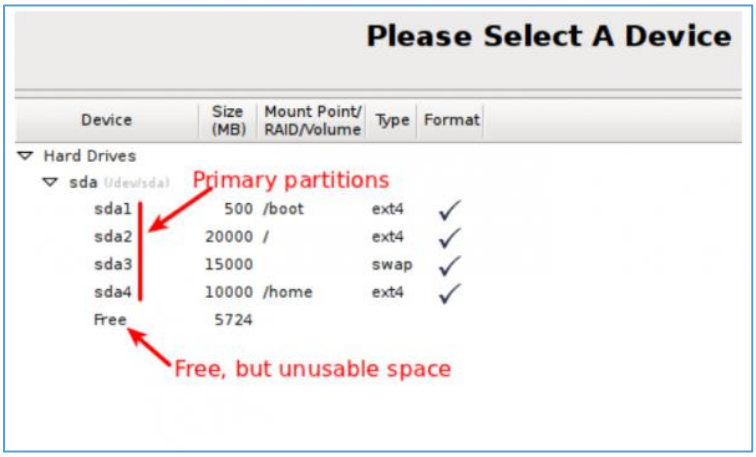


Figure 6: MBR partition numbering in Linux

So if you attempt to create another partition using the free space, the installer will throw up the type of error message shown in Figure 7. The error message will always say, "not enough free space," even when you know that there is space available. And it does not matter whether that free space is 1 MB or 1 GB. It will be unusable.

Figure 7: "Not enough free space on disk" error

To get around the four primary partition limitation of MBR, the smart guys involved came up with the concept of an extended partition. By tagging a partition as an extended partition, it is then possible to create many more partitions under it. Those partitions are called logical partitions. Theoretically, there is no limit to the number of logical partitions that you can create. Note: Only one extended partition may be configured on a single hard drive.

What the concept of extended partition brings to the table is illustrated in Figure 8. You can see that there are three primary partitions – sda1, sda2 and sda3. The fourth partition is an extended partition, which makes it possible to create more (logical) partitions – sda5, sda6 and sda7. Under an extended partition, you can have free space, and that free space will still be usable. So you do not have to allocate all the available free space to the logical partitions of an extended partition.

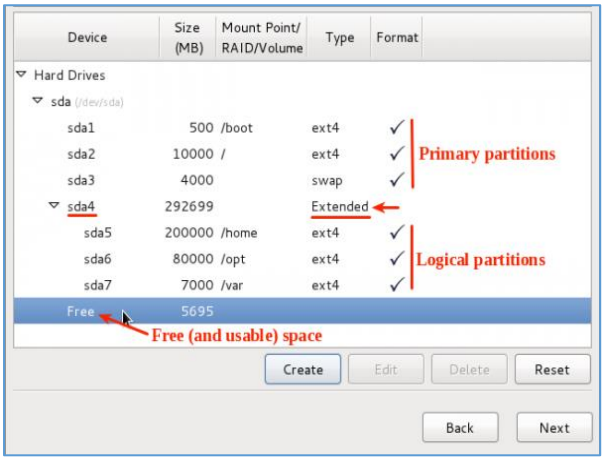


Figure 8: Primary, Extended and Logical partitions in Linux

3. Partitioning the first Hard Drive

3.1. Create BIOS grub volume /dev/sda1 (7.84 MiB)

/dev/sda1 7.00 MiB BIOS Grub (0x00)

/dev/sda2 100.00 MiB EFI boot (0x00) FAT

/boot – The kernel and various other data needed for the system to boot live here.

This partition can usually be pretty small. The data on this partition changes very infrequently and therefore you may not even need the ext3 file systenm on it. (It can use ext2 instead) and having it as a separate partition can facilitate this.

3.2. Create swap volume /dev/sda2 (2.01 GiB)

/dev/sda2 2.01 GiB Linux swap (0x82) Swap swap

swap – the swap partition is used to temporarily store data when the system does not have enough RAM for it's current tasks. As a guide the swap partition should be double the size of system RAM and doesn't need to be bigger than about 2 gigabytes. These days, systems often have more than 2Gb of RAM, but you do still need a swap partition (a system could, technically speaking, function without a swap partition, but I wouldn't recommend it). Considering that disk space is so cheap these days, it is usually not an issue to have a 2Gb (or more) swap partition. The swap partition is never mounted and so does not get a mount point of it's own (i.e. you won't be able to browse the files on your swap partition).

3.3. Create root volume /dev/sda3 (40.00 GiB) with btfrs

/dev/sda3 40 GiB Linux native (0x83) BtrFS /

/ root – This is the root of the file system and will always get it's own partition. Anything that doesn't get a partition of it's own will become part of the root partition.

The default file system for the root partition is Btrfs

3.4. Create volume /dev/sda4 (69.77 GiB) for /home with xfs

Linux native (0x83) XFS /home

/home – this is where the home directories of all users of your system will live. Having this as a separate partition allows you to reinstall the operating system without losing all your personal data. The other advantage is, if you have a multi-user system, a single user won't be able to fill up the whole drive with their data and crash the system (at most only this partition will get filled up).

/usr – most of the packages you install on the system will end up here. Having this as a separate partition will allow you to backup or export this partition to another system easily (if you need to).

/usr/local – many additional packages (those other than Ubuntu core packages, if you're using Ubuntu), will end up here. Some of the software packages you compile from source will also often end up here. You can have this one as a separate partition for similar reasons to */usr*.

/opt – other packages you install and compile from source get installed here, so once again, for similar reasons to /usr and /usr/local – you might want a separate partition for it.

/**var** – the data here usually changes frequently (hence the name). Much of the system log data, package and accounting information resides here. It is a good idea to have it as a separate partition, if someone runs a job that consumes a lot of disk, this area won't be affected and you will still have the information you need (such as logs) to diagnose the problem. The only caveat if you're using Ubuntu is the fact that, the Apache web server will store all it's web content under */var*. So if you're planning to use the system as a web server, it might be prudent to configure a different location for Apache to use.

/var/log – system logs usually live here, and so it may be a good idea to isolate the system logs from other parts of the system, considering how important logs can be do diagnose problems when something goes wrong.

/tmp – this one is pretty self explanatory. It can be used as a temporary space to store files or for programs to write temporary data. Strictly speaking you don't really need a separate partition for this, but it certainly won't hurt.

tmpfs	7.70 GiB	TMPFS	TmpFS	/dev/shm
tmpfs	7.70 GiB	TMPFS	TmpFS	/run
tmpfs	7.70 GiB	TMPFS	TmpFS	/sys/fs/cgroup

1: 15.69 MiB, BIOS GRUB (/dev/sda1)

2: 172.57 MiB, EFI boot (/dev/sda2)

3: 2.11 GiB, Linux swap (/dev/sda3)

4: 40.01 GiB, Linux native (/dev/sda4)

5: 69.52 GiB, Linux native (/dev/sda5)

4. Partitioning the second Hard Drive

The idea is to create a Software depot with LVM (pvcreate, vgcreate, lvcreate)

4.1. Remove All Partitions, Data and Create Empty Disk

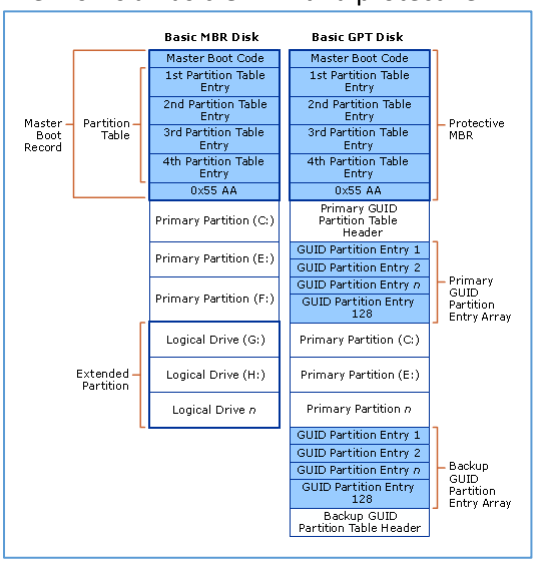
4.1.1. Get the Partitioning Schema (MBR or GPT) with parted

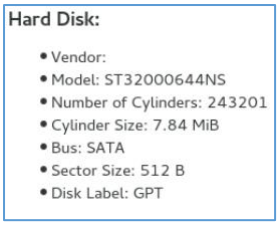
```
linsrv3:~ # parted /dev/sdb print
Model: ATA ST32000644NS (scsi)
Disk /dev/sdb: 2000GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start  End  Size  File system  Name  Flags
```

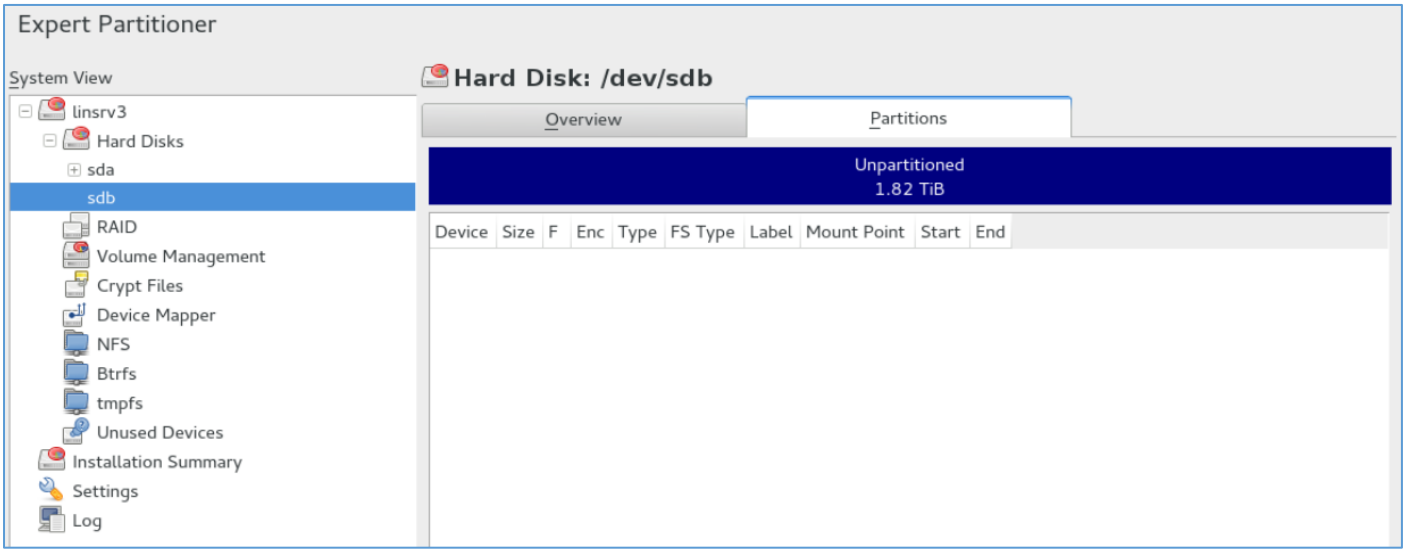
4.1.2. Get the Partitioning Schema (MBR or GPT) with gdisk

```
linsrv3:~ # gdisk -l /dev/sdb
GPT fdisk (gdisk) version 0.8.8

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.
Disk /dev/sdb: 3907029168 sectors, 1.8 TiB
Logical sector size: 512 bytes
Disk identifier (GUID): 54BD6FA4-120E-40E1-BA6E-10400BECFABC
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 3907029134
Partitions will be aligned on 2048-sector boundaries
Total free space is 3907029101 sectors (1.8 TiB)

Number  Start (sector)    End (sector)  Size       Code  Name
```

The Disk is a Basic GPT with a protective MBR.







=> The vgdata is still there??

4.1.3. Setting the Partition Type

Proceeding with YaST2

Proceeding with gdisk

```
linsrv3:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 0.8.8

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.
```

```
Command (? for help): ?
b       back up GPT data to a file
c       change a partition's name
d       delete a partition
i       show detailed information on a partition
l       list known partition types
n       add a new partition
o       create a new empty GUID partition table (GPT)
p       print the partition table
q       quit without saving changes
r       recovery and transformation options (experts only)
s       sort partitions
t       change a partition's type code
v       verify disk
w       write table to disk and exit
x       extra functionality (experts only)
?       print this menu
```

Add the first partition

```
Command (? for help): n
Partition number (1-128, default 1):
First sector (34-3907029134, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-3907029134, default = 3907029134) or {+-}size{KMGTP}: +200G
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'
```

Add the second partition

```
Command (? for help): n
Partition number (2-128, default 2):
First sector (34-3907029134, default = 419432448) or {+-}size{KMGTP}:
Last sector (419432448-3907029134, default = 3907029134) or {+-}size{KMGTP}: +30
0G
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'
```

Add the third partition

```
Command (? for help): n
Partition number (3-128, default 3):
First sector (34-3907029134, default = 1048578048) or {+-}size{KMGTP}:
Last sector (1048578048-3907029134, default = 3907029134) or {+-}size{KMGTP}: +5
00G
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'
```

Add the fourth partition

```
Command (? for help): n
Partition number (4-128, default 4):
First sector (34-3907029134, default = 2097154048) or {+-}size{KMGTP}:
Last sector (2097154048-3907029134, default = 3907029134) or {+-}size{KMGTP}: +5
00G
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'
```

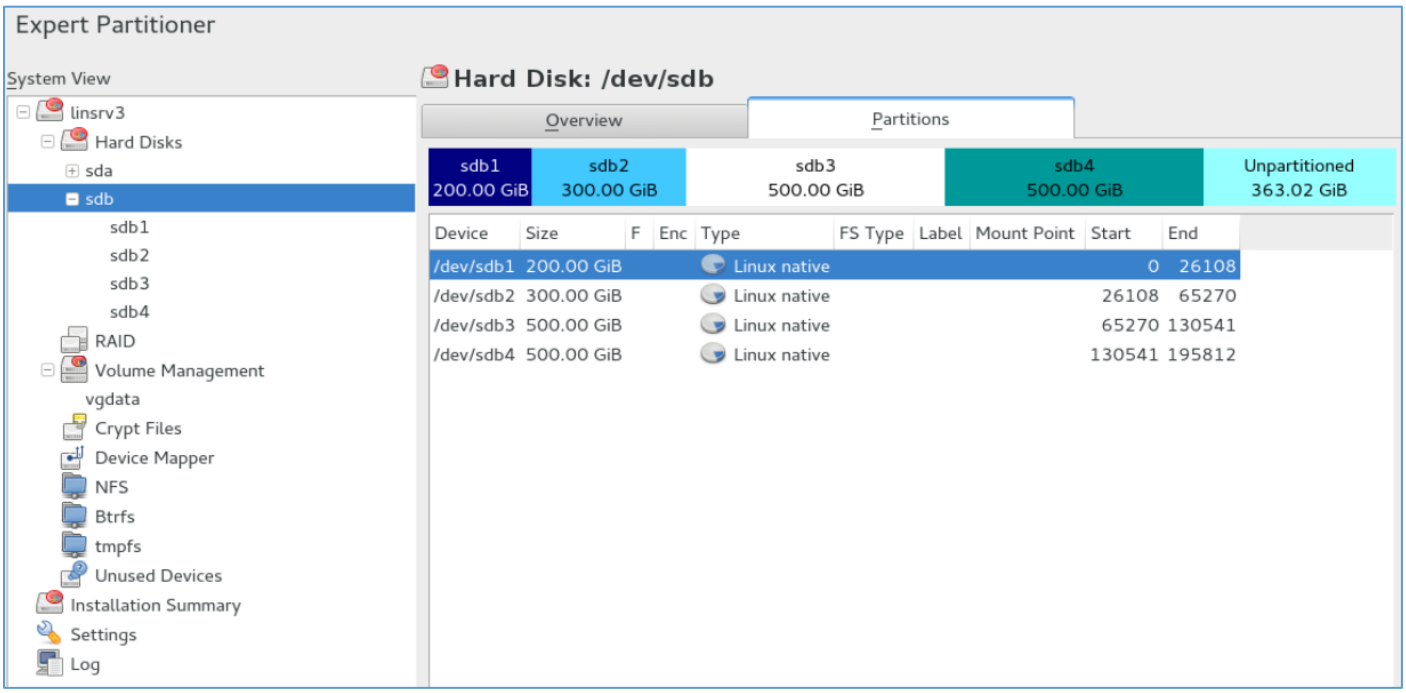
Write the new GUID partition table (GPT)

```
Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!

Do you want to proceed? (Y/N): Y
OK; writing new GUID partition table (GPT) to /dev/sdb.
The operation has completed successfully.
```

```
linsrv3:~ # gdisk -l /dev/sdb
GPT fdisk (gdisk) version 0.8.8

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.
Disk /dev/sdb: 3907029168 sectors, 1.8 TiB
Logical sector size: 512 bytes
Disk identifier (GUID): 54BD6FA4-120E-40E1-BA6E-10400BECFABC
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 3907029134
Partitions will be aligned on 2048-sector boundaries
Total free space is 761301101 sectors (363.0 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048       419432447   200.0 GiB   8300  Linux filesystem
   2       419432448      1048578047   300.0 GiB   8300  Linux filesystem
   3      1048578048      2097154047   500.0 GiB   8300  Linux filesystem
   4      2097154048      3145730047   500.0 GiB   8300  Linux filesystem
```

It is not necessary to format the new partition

```
linsrv3:~ # mkfs.xfs /dev/sdb1
mkfs.xfs: /dev/sdb1 appears to contain an existing filesystem (xfs).
mkfs.xfs: Use the -f option to force overwrite.
```



4.2. Initializing Physical Volumes

Use the pvcreate command to initialize a block device to be used as a physical volume. Initialization is analogous to formatting a file system.

To initialize partitions rather than whole disks: run the pvcreate command on the partition. The following example initializes /dev/sdb1 as an LVM physical volume for later use as part of an LVM logical volume.

```
linsrv3:~ # pvcreate /dev/sdb1
  Can't initialize physical volume "/dev/sdb1" of volume group "vgdata" without
-ff
```

Delete the Volume Group vgdata and the physical Volumes /dev/sdb1 and /dev/sdb3

```
linsrv3:~ # pvdisplay
  --- Physical volume ---
  PV Name               /dev/sdb1
  VG Name               vgdata
  PV Size               500.00 GiB / not usable 4.00 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              127999
  Free PE               127999
  Allocated PE          0
  PV UUID               7LlWmg-V6zQ-uyML-AfaJ-Dqnf-rDH8-viujl3

  --- Physical volume ---
  PV Name               /dev/sdb3
  VG Name               vgdata
  PV Size               500.00 GiB / not usable 4.00 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              127999
  Free PE               127999
  Allocated PE          0
  PV UUID               UltGYR-3gnH-rbFQ-Oz9Y-BBIw-HrFo-hSpLOi
```

```
linsrv3:~ # pvremove /dev/sdb1
  Labels on physical volume "/dev/sdb1" successfully wiped
linsrv3:~ # pvremove /dev/sdb3
  Labels on physical volume "/dev/sdb3" successfully wiped
```

Check with lvmdiskscan

```
linsrv3:~ # lvmdiskscan
  /dev/sda1 [       7.00 MiB]
  /dev/sda2 [       2.01 GiB]
  /dev/sda3 [      23.34 GiB]
  /dev/sda4 [      34.27 GiB]
  /dev/sdb1 [     200.00 GiB]
  /dev/sdb2 [     300.00 GiB]
  /dev/sdb3 [     500.00 GiB]
  /dev/sdb4 [     500.00 GiB]
  0 disks
  8 partitions
  0 LVM physical volume whole disks
  0 LVM physical volumes
```

Create the physical volumes with pvcreate

```
linsrv3:~ # pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
linsrv3:~ # pvcreate /dev/sdb2
  Physical volume "/dev/sdb2" successfully created
linsrv3:~ # pvcreate /dev/sdb3
  Physical volume "/dev/sdb3" successfully created
linsrv3:~ # pvcreate /dev/sdb4
  Physical volume "/dev/sdb4" successfully created
```

Check with pvscan

```
linsrv3:~ # pvscan
  PV /dev/sdb1          lvm2 [200.00 GiB]
  PV /dev/sdb2          lvm2 [300.00 GiB]
  PV /dev/sdb4          lvm2 [500.00 GiB]
  PV /dev/sdb3          lvm2 [500.00 GiB]
  Total: 4 [1.46 TiB] / in use: 0 [0   ] / in no VG: 4 [1.46 TiB]
```

4.3. Creating Volume Groups

To create a volume group from one or more physical volumes, use the vgcreate command. The vgcreate command creates a new volume group by name and adds at least one physical volume to it.
The following command creates a volume group named vgdata that contains physical volumes /dev/sdb1 and /dev/sdb2.

4.3.1. Using vgcreate

```
linsrv3:~ # vgcreate vgdata /dev/sdb1 /dev/sdb2 /dev/sdb3
  Volume group "vgdata" successfully created
linsrv3:~ # vgcreate vgbackup /dev/sdb4
  Volume group "vgbackup" successfully created
```

When physical volumes are used to create a volume group, its disk space is divided into 4MB extents, by default. This extent is the minimum amount by which the logical volume may be increased or decreased in size. Large numbers of extents will have no impact on I/O performance of the logical volume.

4.3.2. Displaying Volume Groups

```
linsrv3:~ # vgdisplay vgdata
  --- Volume group ---
  VG Name               vgdata
  System ID
  Format                lvm2
  Metadata Areas        3
  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                3
  Act PV                3
  VG Size               999.99 GiB
  PE Size               4.00 MiB
  Total PE              255997
  Alloc PE / Size       0 / 0
  Free  PE / Size       255997 / 999.99 GiB
  VG UUID               PK9VWA-7llw-KJic-Dvfk-pVba-ek0x-RWiO4d
```

```
linsrv3:~ # vgdisplay vgbackup
  --- Volume group ---
  VG Name               vgbackup
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               500.00 GiB
  PE Size               4.00 MiB
  Total PE              127999
  Alloc PE / Size       127999 / 500.00 GiB
  Free  PE / Size       0 / 0
  VG UUID               JGBfW3-OsxB-uGX0-1oeS-cTlD-tJQu-OxNtVd
```

4.4. Creating Logical Volumes

To create a logical volume, use the lvcreate command. You can create linear volumes, striped volumes, and mirrored volumes, as described in the following subsections.

If you do not specify a name for the logical volume, the default name lvol# is used where # is the internal number of the logical volume.

The following sections provide examples of logical volume creation for the three types of logical volumes you can create with LVM.

```
linsrv3:~ # lvcreate -L 200G -n lv01 vgdata
  Logical volume "lv01" created.
linsrv3:~ # lvcreate -L 300G -n lv02 vgdata
  Logical volume "lv02" created.
linsrv3:~ # lvcreate -L 500G -n lv03 vgdata
  Volume group "vgdata" has insufficient free space (127997 extents): 128000 req
uired.
linsrv3:~ # lvcreate -l 100%FREE -n lv03 vgdata
  Logical volume "lv03" created.
```

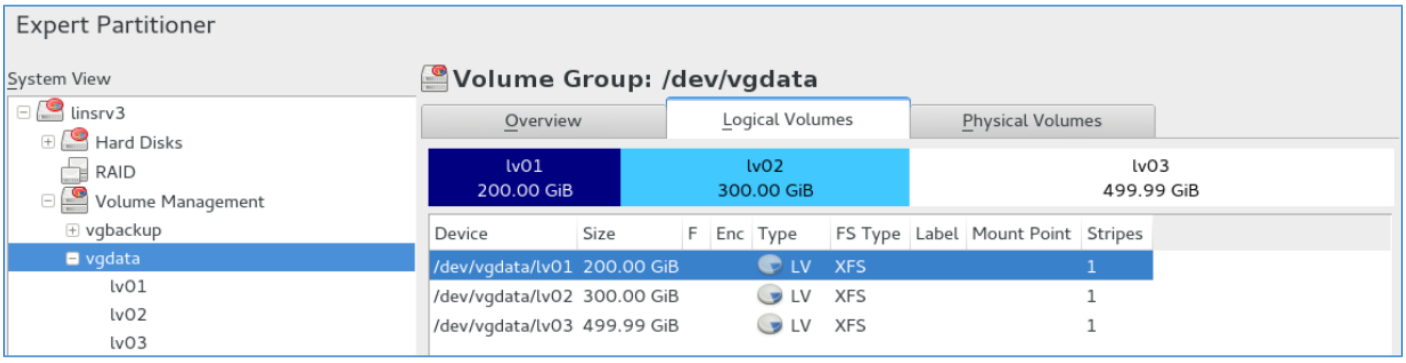
```
linsrv3:~ # vgdisplay vgbackup | grep "Total PE"
  Total PE              127999
linsrv3:~ # lvcreate -l 127999 vgbackup -n lv04
  Logical volume "lv04" created.
```

4.5. Creating File System and mount as normal partitions.

```
linsrv3:~ # mkfs.xfs /dev/vgdata/lv01
meta-data=/dev/vgdata/lv01       isize=256    agcount=4, agsize=13107200 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=52428800, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=25600, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

```
linsrv3:~ # mkfs.xfs /dev/vgdata/lv02
meta-data=/dev/vgdata/lv02       isize=256    agcount=4, agsize=19660800 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=78643200, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=38400, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

```
linsrv3:~ # mkfs.xfs /dev/vgdata/lv03
meta-data=/dev/vgdata/lv03       isize=256    agcount=4, agsize=32767232 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=131068928, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=63998, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```



We have created file system on logical volumes by formatting them. Now we need to create mount point for them. Following command will create new mount points and Mount them as normal partitions. Make sure you select newly created logical volumes.

4.6. Mounting the File System

4.6.1. Per Command Line

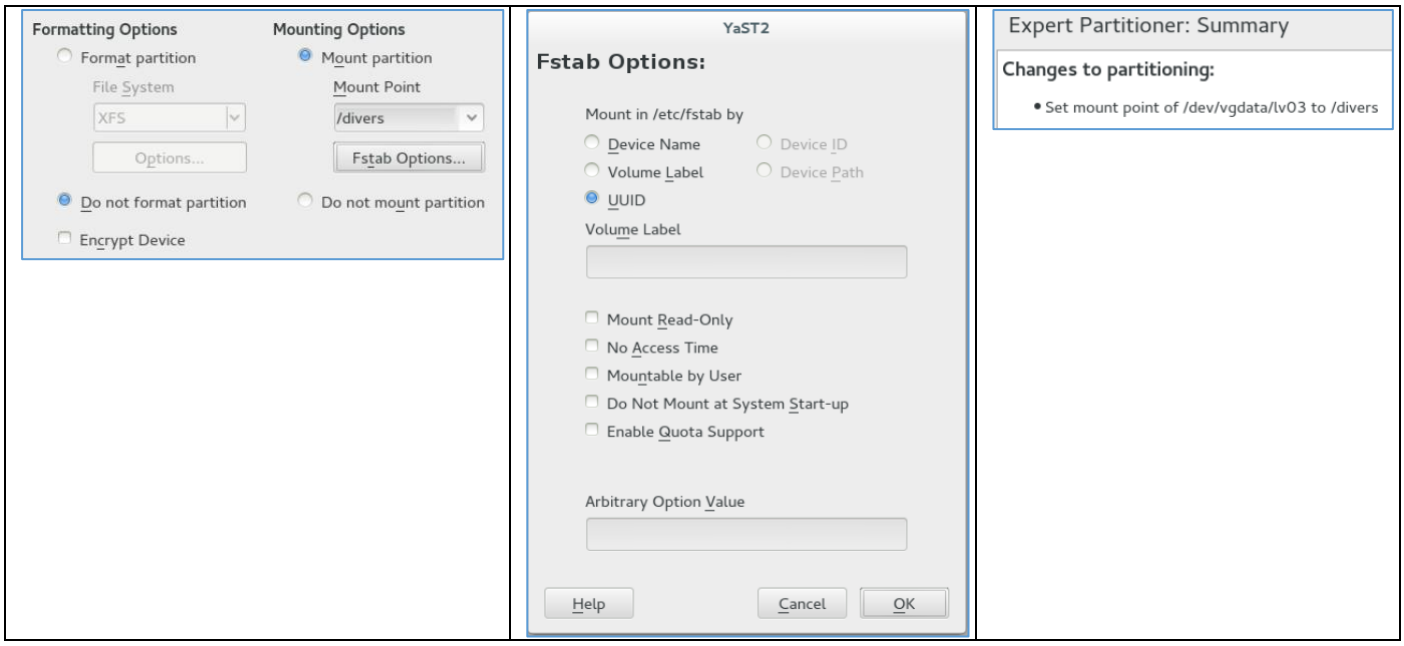
```
linsrv3:~ # mkdir -p /sapcds
linsrv3:~ # mkdir -p /software
linsrv3:~ # mkdir -p /divers
```

```
linsrv3:~ # mount /dev/vgdata/lv01 /sapcds
linsrv3:~ # mount /dev/vgdata/lv02 /software
linsrv3:~ # mount /dev/vgdata/lv03 /divers
```

df -aTh

```
/dev/mapper/vgdata-lv03 xfs                500G    33M   500G    1% /divers
```

4.6.2. YaST > Partitioner



Check in /etc/fstab

```
UUID=41cdb9dd-1819-4ebf-a06d-f29e88b40612 /sapcds              xfs       defaults              1 2
UUID=039dbb6e-23ea-4b2b-b0f5-d8c147a1367d /software            xfs       defaults              1 2
UUID=39a522a3-3861-466c-947f-fa0695282626 /divers              xfs       defaults              1 2
```

df -aTh

```
/dev/mapper/vgdata-lv02 xfs                300G    33M   300G    1% /software
/dev/mapper/vgdata-lv03 xfs                500G    33M   500G    1% /divers
/dev/mapper/vgdata-lv01 xfs                200G    33M   200G    1% /sapcds
```

5. BtrFS Partitioning B-tree

The default file system for the root partition is Btrfs. The root file system is the default subvolume and it is not listed in the list of created subvolumes. As a default Btrfs subvolume, it can be mounted as a normal file system.

IMPORTANT: Btrfs on an Encrypted Root Partition

The default partitioning setup suggests the root partition as Btrfs with /boot being a directory. If you need to have the root partition encrypted in this setup, make sure to use the GPT partition table type instead of the default MSDOS type. Otherwise the GRUB2 boot loader may not have enough space for the second stage loader.

5.1. Suggested Btfrs Subvolumes

/tmp /var/tmp /var/run

Directories with frequently changed content

/var/spool

Contains user data, such as mail

/var/lib

Holds dynamic data libraries and files plus state information pertaining to an application or the system.

By default, subvolumes with the option no copy on write are created for: /var/lib/mariadb, /var/lib/pgsql, and /var/lib/libvirt/images.

/var/log

Contains system and applications' log files which should never be rolled back.

/var/crash

Contains memory dumps of crashed kernels.

/srv

Contains data files belonging to FTP and HTTP servers.

/opt

Contains third party software.

HINT: Size of Btrfs Partition

Because saved snapshots require more disk space, it is recommended to reserve more space for Btrfs partition than for a partition not capable of snapshotting (such as Ext3). Recommended size for a root Btrfs partition with suggested subvolumes is 20GB.

5.2. Subvolumes in Btfrs

A subvolume in btfrs is not the same as an LVM logical volume or a ZFS subvolume. With LVM a logical volume is a block device in its own right (which could for example contain any other filesystem or container like dm-crypt, MD RAID, etc.) This is not the case with btfrs.

A btfrs subvolume is not a block device (and cannot be treated as one) instead, a btfrs subvolume can be thought of as a POSIX file namespace. This namespace can be accessed via the top-level subvolume of the filesystem, or it can be mounted in its own right.

more /etc/fstab

swap	swap	defaults	()	()	
/	btfrs	defaults	()	()	
/boot/grub2/i386-pc	btfrs	subvol=@/boot/grub2/i386-pc	()	()	
/boot/grub2/x86_64-efi	btfrs	subvol=@/boot/grub2/x86_64-efi	()	()	
/opt	btfrs	subvol=@/opt	()	()	
/srv	btfrs	subvol=@/srv	()	()	
/tmp	btfrs	subvol=@/tmp	()	()	
/usr/local	btfrs	subvol=@/usr/local	()	()	
/var/crash	btfrs	subvol=@/var/crash	()	()	
/var/lib/libvirt/images	btfrs	subvol=@/var/lib/libvirt/images	()	()	no copy on write
/var/lib/mailman	btfrs	subvol=@/var/lib/mailman	()	()	
/var/lib/mariadb	btfrs	subvol=@/var/lib/mariadb	()	()	no copy on write
/var/lib/mysql	btfrs	subvol=@/var/lib/mysql	()	()	no copy on write
/var/lib/named	btfrs	subvol=@/var/lib/named	()	()	
/var/lib/pgsql	btfrs	subvol=@/var/lib/pgsql	()	()	no copy on write
/var/log	btfrs	subvol=@/var/log	()	()	
/var/opt	btfrs	subvol=@/var/log	()	()	
/var/spool	btfrs	subvol=@/var/spool	()	()	
/var/tmp	btfrs	subvol=@/var/tmp	()	()	
/.snapshots	btfrs	subvol=@/.snapshots	()	()	
/home	xfs	defaults	1	2	

6. To be redefine

```
Command (m for help): d
Partition number (1,2, default 2):

Partition 2 has been deleted.

Command (m for help): d
Selected partition 1
Partition 1 has been deleted.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Re-reading the partition table failed.: Device or resource busy

The kernel still uses the old table. The new table will be used at the next rebo
ot or after you run partprobe(8) or kpartx(8).
```

```
linsrv3:~ # partprobe
Error: Partition(s) 1, 2 on /dev/sdb have been written, but we have been unable
to inform the kernel of the change, probably because it/they are in use.  As a r
esult, the old partition(s) will remain in use.  You should reboot now before ma
king further changes.
```

Finally Reboot the Server.

```
linsrv3:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 0.8.8

Partition table scan:
  MBR: MBR only
  BSD: not present
  APM: not present
  GPT: present

Found valid MBR and GPT. Which do you want to use?
 1 - MBR
 2 - GPT
 3 - Create blank GPT

Your answer: 3
```

If you are using a whole disk device for your physical volume, the disk must have no partition table. For DOS disk partitions, the partition id should be set to 0x8e using the fdisk or cfdisk command or an equivalent. For whole disk devices only the partition table must be erased, which will effectively destroy all data on that disk. You can remove an existing partition table by zeroing the first sector with the following command:

```
linsrv3:~ # dd if=/dev/zero of=/dev/sdb bs=512 count=1
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.0124527 s, 41.1 kB/s
```