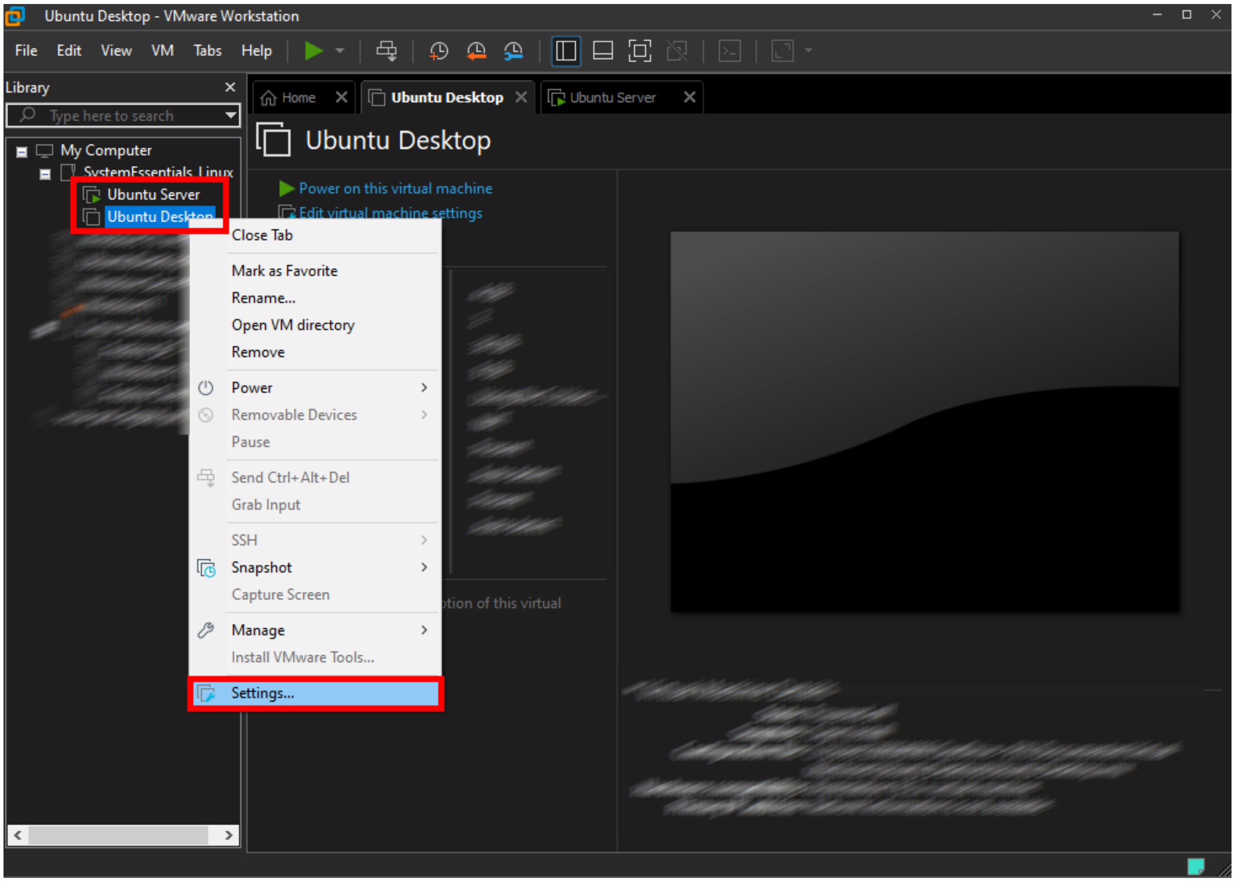
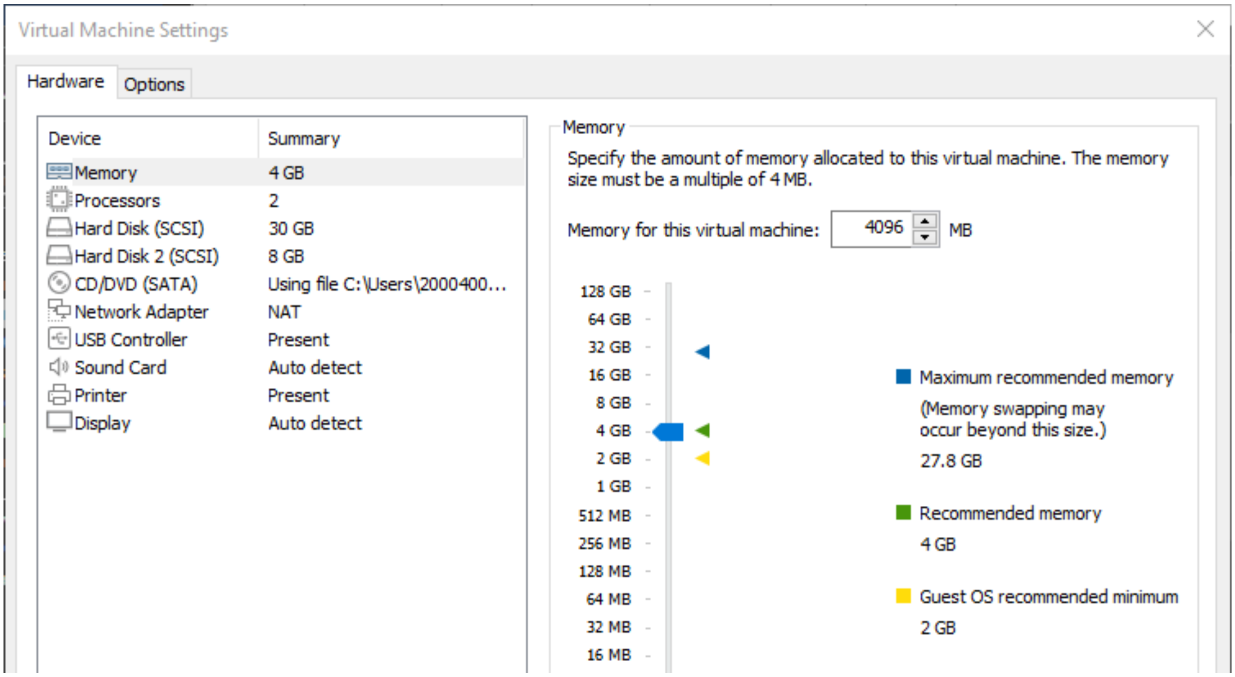
Managing drives and filesystems

Every hard drive that is used in our system is not just plug and play, to use it we need to partition it first. Partitioning means cutting in parts, these parts then receive a certain structure your computer can work with also called a file system. The basics of data storage is almost the same for most operating systems nowadays. When installing an operating system the drive we use, HDD, SSD or NVME, gets partitioned and each partition gets formatted with a certain file system. In Linux some partitions are formatted in a special way, namely the swap partition and Linux Logical Volume Manager (LVM). In our system, HDD's, SSD’s and NVME's are used for permanent storage. RAM and swap are used for temporary storage. For example when running a command, the command gets copied from your drive into RAM to execute it more quickly. This is because our RAM can be read faster than drives by our CPU. However RAM has a smaller capacity due to it’s cost and RAM gets deleted every time our computer is shut down. When our RAM runs out of memory, we can use the swap partition to temporary hold the RAM parts that are not in use and load other data into RAM. The other special partition is the Linux LVM. This logical volume lets us create pools of storage called volume groups. These groups give more flexibility to enlarge or shrink logical volumes than normal drive partitions offer. In our Linux distribution at least 1 partition is necessary, the root partition or "/". But most of the time multiple partitions are created, for example the directories /home, /var or /tmp. Each of these directories get mounted to a mount point under the / partition. When adding files or folders to this mount point, the files and folders get saved on this different partition. These different partitions and mounting of them happens automatically so it’s not visible for the end user. Each of our disk partitions gets a device name when booting Linux, for example sda2. An entry in the file /etc/fstab tells Linux where to mount each partition, which occurs at startup. Note the difference with Windows that everything is mounted under the root (/) and not under a drive letter like C:, D:, … Some drives are automatically mounted to our file system when inserting a removable media. For example, a CD-ROM can get mounted under /media/cdrom or `/run/media/<username>/<cdrom name>` . When this doesn’t occur automatically, an administrator should create the mount point to a folder of his/her choice. Linux is able to work with VFAT, used in USB sticks, handy when exchanging files with a windows system. It also has kernel support for New Technology File System (NTFS), but there are often additional drivers needed to load NTFS.

To start this chapter, we'll need to add drives to our virtual machine first.




Virtual Machine Settings

Hardware | Options

Device	Summary
Memory	4 GB
Processors	2
Hard Disk (SCSI)	30 GB
Hard Disk 2 (SCSI)	8 GB
CD/DVD (SATA)	Using file C:\Users\2000400...
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

Memory

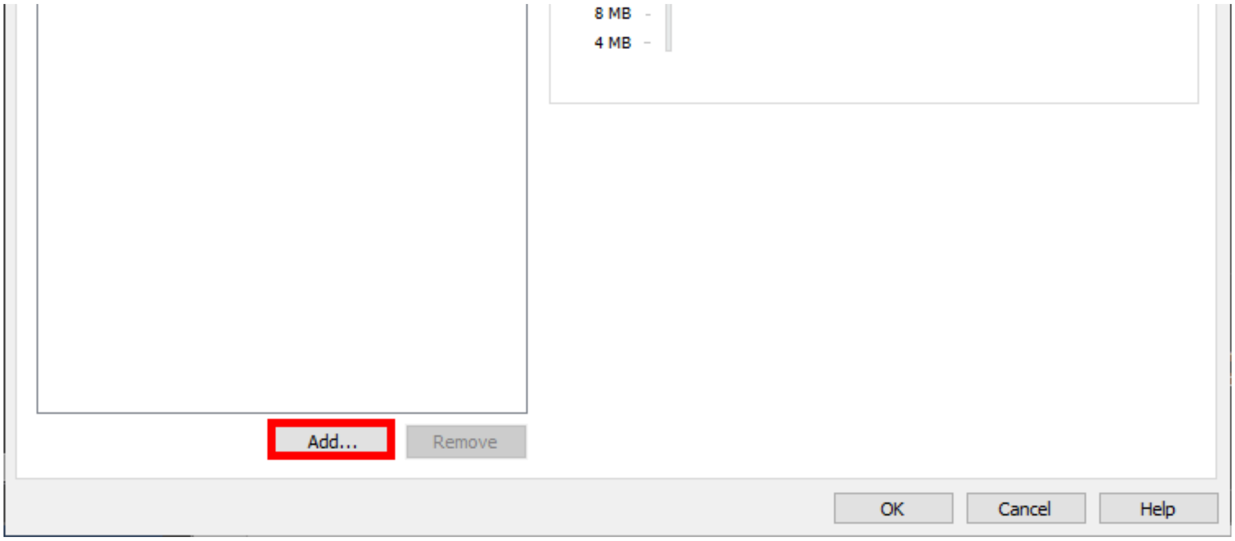
Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.

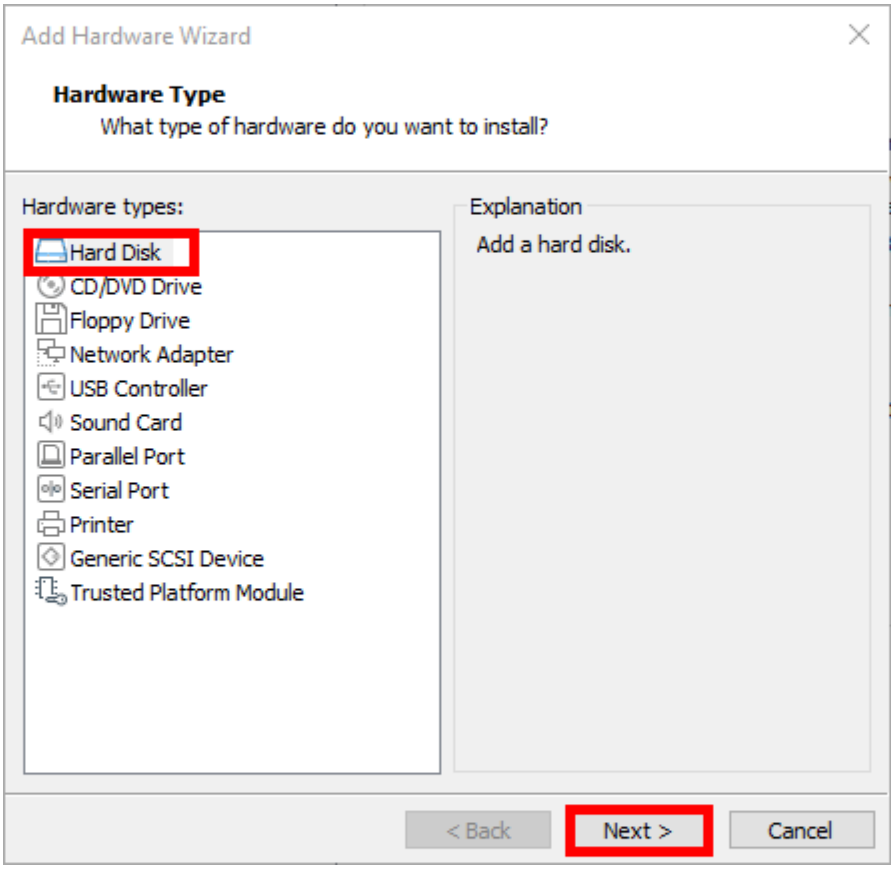
Memory for this virtual machine: 4096 MB

🟦 Maximum recommended memory
(Memory swapping may occur beyond this size.)
27.8 GB

🟩 Recommended memory
4 GB

🟨 Guest OS recommended minimum
2 GB

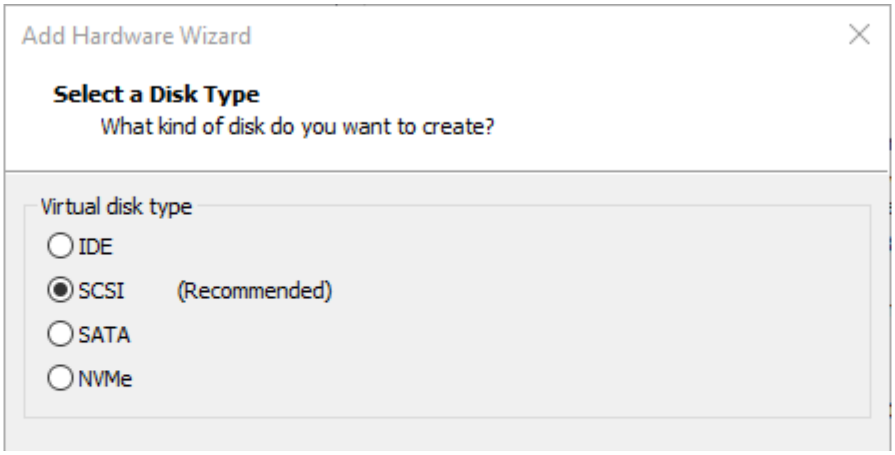



Add Hardware Wizard

Hardware Type

What type of hardware do you want to install?

Add a hard disk.

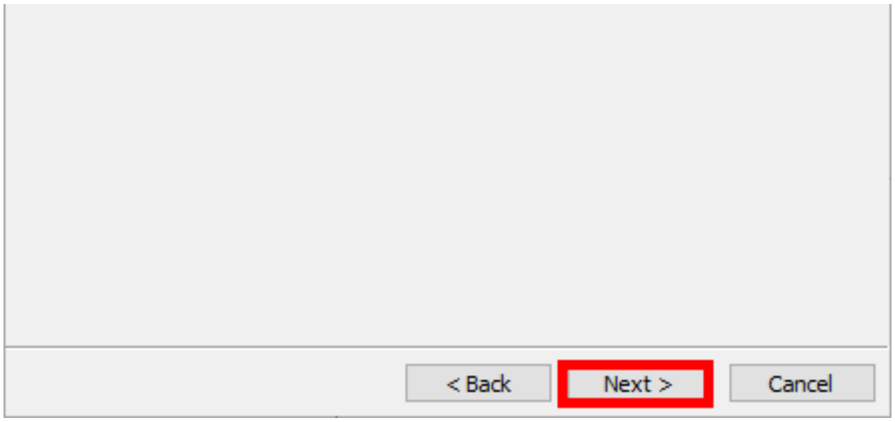

Add Hardware Wizard

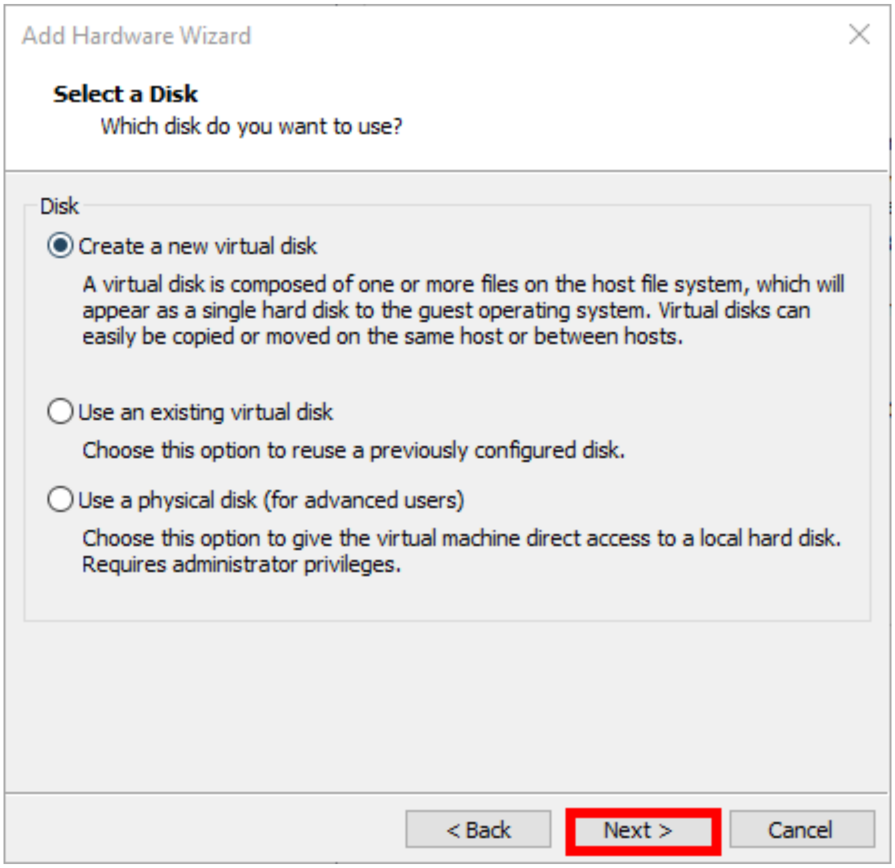
Select a Disk Type

What kind of disk do you want to create?

Virtual disk type

○ IDE
● SCSI (Recommended)
○ SATA
○ NVMe





Select a Disk

Which disk do you want to use?

Disk

● Create a new virtual disk

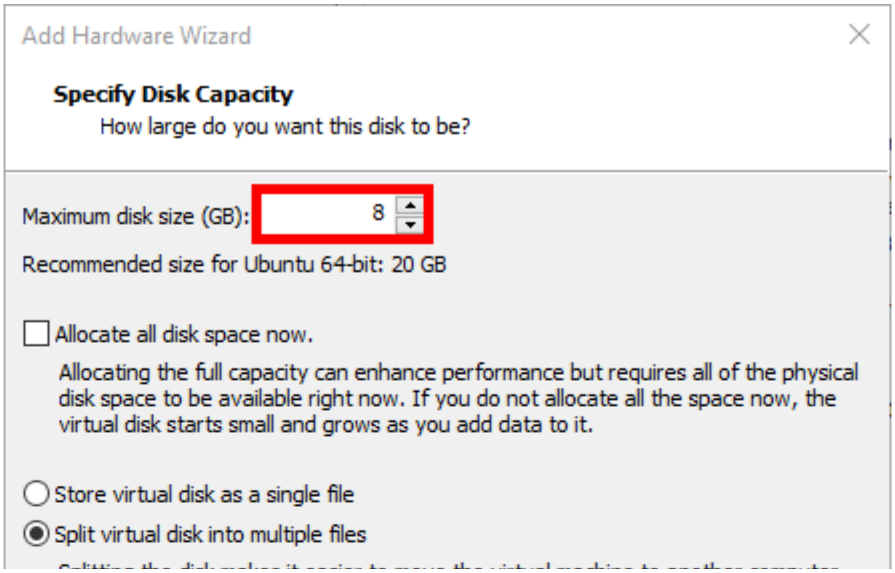
A virtual disk is composed of one or more files on the host file system, which will appear as a single hard disk to the guest operating system. Virtual disks can easily be copied or moved on the same host or between hosts.

○ Use an existing virtual disk

Choose this option to reuse a previously configured disk.

○ Use a physical disk (for advanced users)

Choose this option to give the virtual machine direct access to a local hard disk. Requires administrator privileges.



Specify Disk Capacity

How large do you want this disk to be?

Maximum disk size (GB): 8

Recommended size for Ubuntu 64-bit: 20 GB

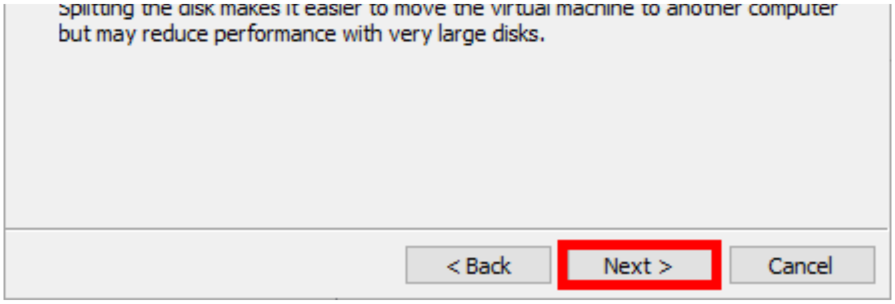
□ Allocate all disk space now.

Allocating the full capacity can enhance performance but requires all of the physical disk space to be available right now. If you do not allocate all the space now, the virtual disk starts small and grows as you add data to it.

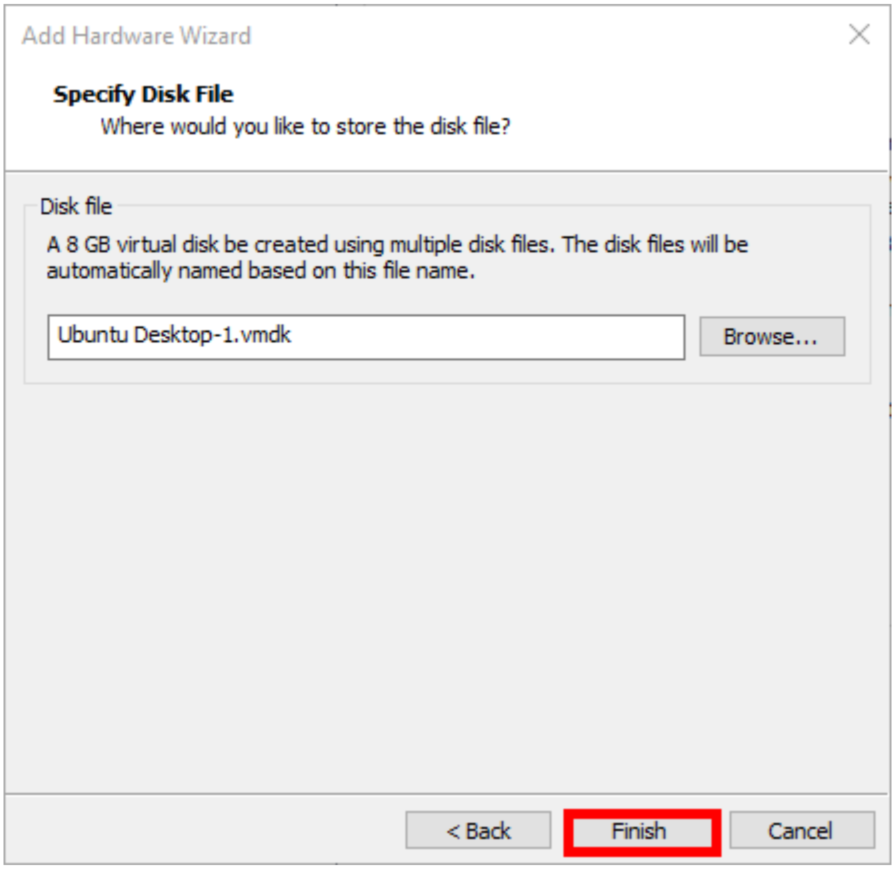
○ Store virtual disk as a single file

● Split virtual disk into multiple files

Splitting the disk makes it easier to move the virtual machine to another computer



Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.



Add Hardware Wizard

Specify Disk File

Where would you like to store the disk file?

Disk file

A 8 GB virtual disk be created using multiple disk files. The disk files will be automatically named based on this file name.

Ubuntu Desktop-1.vmdk

Reboot your Virtual Machine if it was still running, otherwise start it and the drive will be recognised by the kernel and ready to be initialised. Be sure to take a snapshot before continuing this chapter and to double check the steps taken. Be careful and do not repartition the drive on which Linux is installed! A bad entry in the /etc/fstab file could also result in an unbootable Linux.

Understanding partition tables and disk partitions

Traditionally MBR partition tables were used to save the size and layout of partitions. In Linux a lot of tools are available to do this. But nowadays the new standard Global Unique Identifiers, GUID, partition tables are used. Your computer needs to use UEFI for this standard. This change occurred because of the limits of MBR. MBR partitions could be a maximum of 2TB with a maximum of 4 primary partitions or 3 primary and 1 extended partition (which can hold multiple logical partitions). The GUID partitions can get to a maximum of 9,4 ZB (Zettabytes or 10^21^ bytes) with a maximum of 128 partitions. We can use the commands fdisk or gdisk to partition a drive. gdisk gives the possibility to create larger partitions than fdisk, other sub commands to create, delete or change partitions are more or less the same.

bash

```
student@linux-ess:~$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
student@linux-ess:~$
student@linux-ess:~$ sudo gdisk -l /dev/sdb
GPT fdisk (gdisk) version 1.0.8

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

Found valid GPT with protective MBR; using GPT.
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: VMware Virtual S
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 4B226FAA-BFB1-4A6F-9F76-CA70124E6336
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 16777149 sectors (8.0 GiB)

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

Every SCSI, SATA or USB device gets represented by sd? (sda, sdb, sdc, …). With MBR these devices can have a maximum of 16 subdivisions (sdc, sdc1 -> sdc15) this means there is a maximum of 15 partitions with MBR. When using GPT there is a maximum of 128 partitions (sdd, sdd1 -> sdd127). With MBR, a drive can have 4 (primary) partitions maximum. If you need more than 4 partitions, you'll need to use extended partition(s) with logical partitions. The first drive mostly shows as /dev/sda. As said before at least one partition is created when installing Linux, this partion is used as a Linux LVM physical partition where other logical partitions can be created.

bash

```
student@linux-ess:~$ sudo fdisk -l /dev/sda
[sudo] password for student:
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VMware Virtual S
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: gpt
Disk identifier: A9E92175-1433-43DC-968D-95C5E18A2105

Device       Start      End  Sectors  Size Type
/dev/sda1     2048     4095     2048    1M BIOS boot
/dev/sda2     4096  3719167  3715072  1.8G Linux filesystem
/dev/sda3  3719168 41940991 38221824 18.2G Linux filesystem
student@linux-ess:~$
student@linux-ess:~$ sudo gdisk -l /dev/sda
GPT fdisk (gdisk) version 1.0.8

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

Found valid GPT with protective MBR; using GPT.
Disk /dev/sda: 41943040 sectors, 20.0 GiB
Model: VMware Virtual S
Sector size (logical/physical): 512/512 bytes
```

```
Disk identifier (GUID): A9E92175-1433-43DC-968D-95C5E18A2105
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 41943006
Partitions will be aligned on 2048-sector boundaries
Total free space is 4029 sectors (2.0 MiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048            4095   1024.0 KiB  EF02
   2            4096         3719167   1.8 GiB     8300
   3         3719168        41940991   18.2 GiB    8300
```

Looking at our sda drive we see it partitioned into /boot of +- 1GB. The * indicates this partition is bootable. The rest of the drive is a physical LVM partition. This one is used to create logical volumes. With lsblk we see all available drives and their partitions.

bash

```
student@linux-ess:~$ lsblk
$NAME                      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0                       7:0    0 63.2M  1 loop /snap/core20/1623
loop1                       7:1    0   48M  1 loop /snap/snapd/16778
loop2                       7:2    0 79.9M  1 loop /snap/lxd/22923
loop3                       7:3    0   62M  1 loop /snap/core20/1587
loop4                       7:4    0  103M  1 loop /snap/lxd/23541
loop5                       7:5    0   47M  1 loop /snap/snapd/16292
sda                         8:0    0   20G  0 disk
├─sda1                      8:1    0    1M  0 part
├─sda2                      8:2    0  1.8G  0 part /boot
└─sda3                      8:3    0 18.2G  0 part
  └─ubuntu--vg-ubuntu--lv 253:0    0   10G  0 lvm  /
sdb                         8:16   0    8G  0 disk
sr0                        11:0    1  1.4G  0 rom
```

Adding a disk with one partition

Next up, we'll partition the new drive and install a file system. Afterwards we'll be able to mount the new partition to a folder in Linux. The easiest method is to use the full drive space for one partition. It is possible to add multiple partitions, you'll need to add a file system to each partition afterwards and each partition needs to be mounted separately. If a partition of a drive is mounted, like a USB stick, and you want to repartition it, you'll need to unmount it first. We'll now start with creating a partition and installing a file system on our extra drive. If a mistake is made while working with the fdisk/gdisk command, finish the current operation and press q to quit without saving.

First up, check the device name of our newly added drive by using the lsblk command

```bash
student@linux-ess:~$ lsblk
$NAME                        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0                          7:0    0 63.2M  1 loop /snap/core20/1623
loop1                          7:1    0   48M  1 loop /snap/snapd/16778
loop2                          7:2    0 79.9M  1 loop /snap/lxd/22923
loop3                          7:3    0   62M  1 loop /snap/core20/1587
loop4                          7:4    0  103M  1 loop /snap/lxd/23541
loop5                          7:5    0   47M  1 loop /snap/snapd/16292
sda                            8:0    0   20G  0 disk
├─sda1                         8:1    0    1M  0 part
├─sda2                         8:2    0  1.8G  0 part /boot
└─sda3                         8:3    0 18.2G  0 part
  └─ubuntu--vg-ubuntu--lv    253:0    0   10G  0 lvm  /
sdb                            8:16   0    8G  0 disk                   # C
sr0                           11:0    1  1.4G  0 rom
```

In this case we'll need to use /dev/sdb. Now start by using the fdisk or gdisk command and the drive we want to use.

```bash
student@linux-ess:~$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.37.2).
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 0x8b8ca071.

Command (m for help):
```

bash

```
student@linux-ess:~$ sudo gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.8

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

Found valid GPT with protective MBR; using GPT.

Command (? for help):
```

We'll fist check if our drive isn't already formatted. Press p to check for partitions.

bash

```
# FDISK
Command (m for help): p
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Disk model: VMware Virtual S
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: 0x8b8ca071

Command (m for help):
```

bash

```
# GDISK
Command (? for help): p
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: VMware Virtual S
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 4B226FAA-BFB1-4A6F-9F76-CA70124E6336
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 16777149 sectors (8.0 GiB)

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

Command (? for help):
```

If a partition would be present, we’ll need to delete it before going on. You could do this by pressing d. It will then tell you what partition is selected, pressing enter will delete this partition. We are now ready to create a new partition, do this by pressing n. We’ll now be prompted to choose a primary (p), or extended (e) partition with fdisk, gdisk doesn't ask for this. As this is our first partition, choose the primary partition by pressing p. Afterwords enter the partition number, again as this is our first partition, we’ll use number 1. Do this by pressing 1 and pressing enter. Next we'll be prompted where the first sector should start, we’ll use the default value, so just press enter. Now it’s time to set the size of the partition, you could enter a number, but we’ll use the full size of our drive. Enter the number of the last sector or just press enter. Now we can check our newly created partition by pressing p. If everything is as expected, press w to write or save the changes to the partition table.

bash

```
# FDISK
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
```

```
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-16777215, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-16777215, default 167

Created a new partition 1 of type 'Linux' and of size 8 GiB.

Command (m for help): p
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Disk model: VMware Virtual S
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: 0x8b8ca071

Device     Boot Start      End  Sectors Size Id Type
/dev/sdb1        2048 16777215 16775168   8G 83 Linux

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

```bash
# GDISK
Command (? for help): n
Partition number (1-128, default 1): 1
First sector (34-16777182, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-16777182, default = 16777182) or {+-}size{KMGTP}:
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'

Command (? for help): p
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: VMware Virtual S
Sector size (logical/physical): 512/512 bytes
```

```
Disk identifier (GUID): 4B226FAA-BFB1-4A6F-9F76-CA70124E6336
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 2014 sectors (1007.0 KiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048        16777182   8.0 GiB     8300  Linux filesystem

Command (? for help): w

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

Do you want to proceed? (Y/N): Y
OK; writing new GUID partition table (GPT) to /dev/sdb.
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.
```

If this should fail its probably because the drive was still mounted. Unmount the drive and use the partprobe command to save the changes to the partition table. If that still fails, reboot your computer and try again.

```bash
student@linux-ess:~$ sudo partprobe /dev/sdb
```

The new partition still is not ready for use. We need to install a file system on it as mentioned before. Use the mkfs command to accomplish this. The standard mkfs command creates a ext2 file system, normally you would like a journaling system like ext3/ ext4. Change the mkfs command with parameter -t to get this. Another option is to use the mkfs.ext4 command, this is a shorter option. We could also use a xfs-file system.

```bash
student@linux-ess:~$ sudo mkfs -t ext4 /dev/sdb1
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 2096896 4k blocks and 524288 inodes
Filesystem UUID: f2fee344-9305-46d7-9942-339fd434dd8d
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

student@linux-ess:~$ sudo mkfs.ext4 /dev/sdb1
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 2096896 4k blocks and 524288 inodes
Filesystem UUID: f2fee344-9305-46d7-9942-339fd434dd8d
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

student@linux-ess:~$ sudo mkfs -t xfs /dev/sdb1
meta-data=/dev/sdb1              isize=512    agcount=4, agsize=524224 bl
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=
         =                       reflink=1    bigtime=0 inobtcount=0
data     =                       bsize=4096   blocks=2096896, 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

student@linux-ess:~$ sudo mkfs.xfs /dev/sdb1
meta-data=/dev/sdb1              isize=512    agcount=4, agsize=524224 bl
```

```
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=
         =                       reflink=1    bigtime=0 inobtcount=0
data     =                       bsize=4096   blocks=2096896, 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
```

Now that we created a partition and created a file system on it, it's time to mount our newly created partition to a mount point. Use the mount command to do this.

bash

```
student@linux-ess:~$ sudo mkdir /mnt/test
student@linux-ess:~$ sudo mount /dev/sdb1 /mnt/test
student@linux-ess:~$ sudo df -h /mnt/test
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb1       7.8G   24K  7.4G   1% /mnt/test
student@linux-ess:~$ sudo mount | grep sdb1
/dev/sdb1 on /mnt/test type ext4 (rw,relatime)
```

With the df command we see that /dev/sdb1 is mounted to the folder /mnt/test. We can also see it provides 7.8GB of memory. The mount command shows all mounted partitions, with grep we filter to see only our new partition. When the partition is no longer in needed, we can unmount it with the umount command

bash

```
student@linux-ess:~$ sudo umount /dev/sdb1
```

With the mount command, the partition is temporarily mounted and gets unmounted when the pc gets rebooted. When we want the partition to be remounted on startup we'll need to add an entry for it in the /etc/fstab file.

bash

```
student@linux-ess:~$ sudo nano /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name device
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/ubuntu-vg/ubuntu-lv during curtin installation
/dev/disk/by-id/dm-uuid-LVM-cPT0cPGoZhK91HrSeQ2ByYSCtWdNqE7Co7jRC0O2XWz9d
# /boot was on /dev/sda2 during curtin installation
/dev/disk/by-uuid/0b189ed9-2d70-4955-9e36-66ad45dbbee7 /boot ext4 default
/dev/sdb1 /mnt/test ext4 defaults 0 2
```

In this example /dev/sdb1 gets mounted to the folder /mnt/test with ext4 as its file system. The word defaults means that it gets mounted with default options (rw, auto, …). The two numbers at the end stand for: the first one tells the system it does not need to make backup files of this file system with the help of the dump command. This command is barely used nowadays, but the number field is still available. The second one indicates the order on which the system checks this mount. This value is 1 for the root file system and 2 for others. If you do not want to check this partition at boot time set this field to 0. Now that this partition is added to the /etc/fstab file, it will be mounted every time your system boots.

adding a disk with multiple partitions

We'll now start again with our drive and try to create multiple partitions. The partitions we want are:

- sdb1 and sbd2: 500 MB
- sdb3: 300MB
- sdb5: 350 MB
- sdb6: 400 MB

Sdb4, which is missing in our list, is an extended partition when using fdisk, which takes all remaining disk space (after sdb1, sdb2 and sdb3).
sdb5 and sdb6 are logical partitions within the extended partition (sdb4).
When using gdisk this is not needed, but we will use the same numbers to keep consistency.
First, unmount all mounted partitions of this drive. And if they were mounted automatically with an entry in the /etc/fstab file, you'll also need to remove those entries from /etc/fstab.

bash

```
student@linux-ess:~$ sudo umount /dev/sdb1
student@linux-ess:~$ sudo nano /etc/fstab     #to remove the sdb1 entry
```

Now we go back to our fdisk or gdisk command.

bash

```
student@linux-ess:~$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help):
```

bash

```
student@linux-ess:~$ sudo gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.8

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

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

```
Command (? for help):
```

Check for existing partitions, we know there is one, by pressing p. We'll delete this partition by entering d, checking the selected partition and if correct, pressing enter.

```bash
# FDISK
Command (m for help): p
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Disk model: VMware Virtual S
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: 0x8b8ca071

Device     Boot Start      End  Sectors Size Id Type
/dev/sdb1        2048 16777215 16775168   8G 83 Linux

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

Command (m for help):
```

```bash
# GDISK
Command (? for help): p
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: VMware Virtual S
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 4B226FAA-BFB1-4A6F-9F76-CA70124E6336
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
```

```
Total free space is 2014 sectors (1007.0 KiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048        16777182   8.0 GiB     8300  Linux filesystem

Command (? for help): d
Using 1

Command (? for help):
```

Now, we'll create new partitions by entering n, for sdb1, sdb2 and sdb3 we'll use primary partitions. So use p, when using fdisk. Enter the number of the partition, 1. We now need to enter the amount to allocate to the partition. Do this by entering a + followed by the amount. This is asked in bytes, but its easier to use K, M, G for bigger numbers. Our first partition needs to be 500MB, so we enter +500M. If you did not remove the previous partition you'll get a notification the partion has an ext4 signature. You'll need to enter *y* to the question if this can be removed.

```bash
# FDISK
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-16777215, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-16777215, default 167

Created a new partition 1 of type 'Linux' and of size 500 MiB.
Partition #1 contains a ext4 signature.

Do you want to remove the signature? [Y]es/[N]o: y

The signature will be removed by a write command.

Command (m for help):
```

```bash
# GDISK
Command (? for help): n
Partition number (1-128, default 1): 1
First sector (34-16777182, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-16777182, default = 16777182) or {+-}size{KMGTP}: +500M
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'

Command (? for help):
```

Create the next 2 partitions the same way, sdb2 with 500MB and sdb3 with 300MB.

```bash
# FDISK
```

```
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 (1026048-16777215, default 1026048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (1026048-16777215, default

Created a new partition 2 of type 'Linux' and of size 500 MiB.

Command (m for help): n
Partition type
   p   primary (2 primary, 0 extended, 2 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (3,4, default 3): 3
First sector (2050048-16777215, default 2050048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2050048-16777215, default

Created a new partition 3 of type 'Linux' and of size 300 MiB.

Command (m for help):
```

bash

```
# GDISK
Command (? for help): n
Partition number (2-128, default 2): 2
First sector (34-16777182, default = 1026048) or {+-}size{KMGTP}:
Last sector (1026048-16777182, default = 16777182) or {+-}size{KMGTP}: +5
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'

Command (? for help): n
Partition number (3-128, default 3): 3
First sector (34-16777182, default = 2050048) or {+-}size{KMGTP}:
Last sector (2050048-16777182, default = 16777182) or {+-}size{KMGTP}: +3
```

```
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'

Command (? for help):
```

For the fourth partition, we change the partition type to extended when using fdisk as mentioned before. With this extended partition we will claim all the free space that is left, so we can create multiple logical partitions within this extended partition (space).

```bash
# FDISK
Command (m for help): n
Partition type
   p   primary (3 primary, 0 extended, 1 free)
   e   extended (container for logical partitions)
Select (default e): e

Selected partition 4
First sector (2664448-16777215, default 2664448):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2664448-16777215, default

Created a new partition 4 of type 'Extended' and of size 6.7 GiB.

Command (m for help):
```

We do not make an extended partition with gdisk, we can create up to 128 partitions, so no need to create an extended one! Now create partition 5 with 350MB and 6 with 400MB as before, notice we do not need to choose a partition type this time. This is not possible as all the partition space is used in fdisk. Our fdisk command knows there is an extended partition and that it needs to use this one for any new partitions.

```bash
# FDISK
Command (m for help): n
```

```
All primary partitions are in use.
Adding logical partition 5
First sector (2666496-16777215, default 2666496):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2666496-16777215, default

Created a new partition 5 of type 'Linux' and of size 350 MiB.

Command (m for help): n
All primary partitions are in use.
Adding logical partition 6
First sector (3385344-16777215, default 3385344):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (3385344-16777215, default

Created a new partition 6 of type 'Linux' and of size 400 MiB.

Command (m for help):
```

With gdisk we will skip partition 4 to keep the same numbers as with the fdisk demo. Otherwise we wouldn't do that.

bash

```
# GDISK
Command (? for help): n
Partition number (4-128, default 4): 5
First sector (34-16777182, default = 2664448) or {+-}size{KMGTP}:
Last sector (2664448-16777182, default = 16777182) or {+-}size{KMGTP}: +3
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'

Command (? for help): n
Partition number (4-128, default 4): 6
First sector (34-16777182, default = 3381248) or {+-}size{KMGTP}:
Last sector (3381248-16777182, default = 16777182) or {+-}size{KMGTP}: +4
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'
```

```
Command (? for help):
```

Check the partition table by entering p. If we want to check if there is any free space on our drive we can do so by entering F in fdisk, gdisk shows this when pressing p.

```bash
# FDISK
Command (m for help): p
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Disk model: VMware Virtual S
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: 0x8b8ca071

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sdb1          2048  1026047  1024000  500M 83 Linux
/dev/sdb2       1026048  2050047  1024000  500M 83 Linux
/dev/sdb3       2050048  2664447   614400  300M 83 Linux
/dev/sdb4       2664448 16777215 14112768  6.7G  5 Extended
/dev/sdb5       2666496  3383295   716800  350M 83 Linux
/dev/sdb6       3385344  4204543   819200  400M 83 Linux

Filesystem/RAID signature on partition 1 will be wiped.

Command (m for help): F
Unpartitioned space /dev/sdb: 5.99 GiB, 6436159488 bytes, 12570624 sector
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes

  Start      End  Sectors Size
4206592 16777215 12570624   6G

Command (m for help):
```

```bash
# GDISK
Command (? for help): p
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: VMware Virtual S
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 4B226FAA-BFB1-4A6F-9F76-CA70124E6336
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 12578749 sectors (6.0 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         1026047   500.0 MiB   8300  Linux filesystem
   2         1026048         2050047   500.0 MiB   8300  Linux filesystem
   3         2050048         2664447   300.0 MiB   8300  Linux filesystem
   5         2664448         3381247   350.0 MiB   8300  Linux filesystem
   6         3381248         4200447   400.0 MiB   8300  Linux filesystem

Command (? for help):
```

We can see that all partition types are Linux at the moment, we'll change some to swap, FAT32 and Linux LVM. Do this by entering t and the number of the file system we want. To get a list of all file systems enter l, we need to check the number of swap (82), FAT32 (0c) and Linux LVM (8e).

```bash
# FDISK
Command (m for help): l

00 Empty            24 NEC DOS          81 Minix / old Lin  bf Solaris
01 FAT12            27 Hidden NTFS Win  82 Linux swap / So  c1 DRDOS/sec
02 XENIX root       39 Plan 9           83 Linux            c4 DRDOS/sec
03 XENIX usr        3c PartitionMagic   84 OS/2 hidden or   c6 DRDOS/sec
04 FAT16 <32M       40 Venix 80286      85 Linux extended   c7 Syrinx
05 Extended         41 PPC PReP Boot    86 NTFS volume set  da Non-FS dat
06 FAT16            42 SFS              87 NTFS volume set  db CP/M / CTO
```

```
07 HPFS/NTFS/exFAT  4d QNX4.x           88 Linux plaintext  de Dell Utili
08 AIX              4e QNX4.x 2nd part  8e Linux LVM        df BootIt
09 AIX bootable     4f QNX4.x 3rd part  93 Amoeba           e1 DOS access
0a OS/2 Boot Manag  50 OnTrack DM       94 Amoeba BBT       e3 DOS R/O
0b W95 FAT32        51 OnTrack DM6 Aux  9f BSD/OS           e4 SpeedStor
0c W95 FAT32 (LBA)  52 CP/M             a0 IBM Thinkpad hi  ea Linux exte
0e W95 FAT16 (LBA)  53 OnTrack DM6 Aux  a5 FreeBSD          eb BeOS fs
0f W95 Ext´d (LBA)  54 OnTrackDM6       a6 OpenBSD          ee GPT
10 OPUS             55 EZ-Drive         a7 NeXTSTEP         ef EFI (FAT-1
11 Hidden FAT12     56 Golden Bow       a8 Darwin UFS       f0 Linux/PA-R
12 Compaq diagnost  5c Priam Edisk      a9 NetBSD           f1 SpeedStor
14 Hidden FAT16 <3  61 SpeedStor        ab Darwin boot      f4 SpeedStor
16 Hidden FAT16     63 GNU HURD or Sys  af HFS / HFS+       f2 DOS second
17 Hidden HPFS/NTF  64 Novell Netware   b7 BSDI fs          fb VMware VMF
18 AST SmartSleep   65 Novell Netware   b8 BSDI swap        fc VMware VMK
1b Hidden W95 FAT3  70 DiskSecure Mult  bb Boot Wizard hid  fd Linux raid
1c Hidden W95 FAT3  75 PC/IX            bc Acronis FAT32 L  fe LANstep
1e Hidden W95 FAT1  80 Old Minix        be Solaris boot     ff BBT

Aliases:
   linux          - 83
   swap           - 82
   extended       - 05
   uefi           - EF
   raid           - FD
   lvm            - 8E
   linuxex        - 85

Command (m for help): t
Partition number (1-6, default 6): 2
Hex code or alias (type L to list all): 82

Changed type of partition 'Linux' to 'Linux swap / Solaris'.

Command (m for help): t
Partition number (1-6, default 6): 5
Hex code or alias (type L to list all): c

Changed type of partition 'Linux' to 'W95 FAT32 (LBA)'.
```

```
Command (m for help): t
Partition number (1-6, default 6): 6
Hex code or alias (type L to list all): 8e

Changed type of partition 'Linux' to 'Linux LVM'.
```

bash

```
# GDISK
Command (? for help): l
Type search string, or <Enter> to show all codes:
0700 Microsoft basic data                0701 Microsoft Storage Replica
0702 ArcaOS Type 1                       0c01 Microsoft reserved
2700 Windows RE                          3000 ONIE boot
3001 ONIE config                         3900 Plan 9
4100 PowerPC PReP boot                   4200 Windows LDM data
4201 Windows LDM metadata                4202 Windows Storage Spaces
7501 IBM GPFS                            7f00 ChromeOS kernel
7f01 ChromeOS root                       7f02 ChromeOS reserved
8200 Linux swap                          8300 Linux filesystem
8301 Linux reserved                      8302 Linux /home
8303 Linux x86 root (/)                  8304 Linux x86-64 root (/)
8305 Linux ARM64 root (/)                8306 Linux /srv
8307 Linux ARM32 root (/)                8308 Linux dm-crypt
8309 Linux LUKS                          830a Linux IA-64 root (/)
830b Linux x86 root verity               830c Linux x86-64 root verity
830d Linux ARM32 root verity             830e Linux ARM64 root verity
830f Linux IA-64 root verity             8310 Linux /var
8311 Linux /var/tmp                      8312 Linux user\'s home
8313 Linux x86 /usr                      8314 Linux x86-64 /usr
8315 Linux ARM32 /usr                    8316 Linux ARM64 /usr
8317 Linux IA-64 /usr                    8318 Linux x86 /usr verity
Press the <Enter> key to see more codes, q to quit:
8319 Linux x86-64 /usr verity            831a Linux ARM32 /usr verity
831b Linux ARM64 /usr verity             831c Linux IA-64 /usr verity
8400 Intel Rapid Start                   8401 SPDK block device
8500 Container Linux /usr                8501 Container Linux resizable r
8502 Container Linux /OEM customization  8503 Container Linux root on RAI
8e00 Linux LVM                           a000 Android bootloader
```

```
a001 Android bootloader 2                a002 Android boot 1
a003 Android recovery 1                  a004 Android misc
a005 Android metadata                    a006 Android system 1
a007 Android cache                       a008 Android data
a009 Android persistent                  a00a Android factory
a00b Android fastboot/tertiary           a00c Android OEM
a00d Android vendor                      a00e Android config
a00f Android factory (alt)               a010 Android meta
a011 Android EXT                         a012 Android SBL1
a013 Android SBL2                        a014 Android SBL3
a015 Android APPSBL                      a016 Android QSEE/tz
a017 Android QHEE/hyp                    a018 Android RPM
a019 Android WDOG debug/sdi              a01a Android DDR
a01b Android CDT                         a01c Android RAM dump
a01d Android SEC                         a01e Android PMIC
Press the <Enter> key to see more codes, q to quit:
a01f Android misc 1                      a020 Android misc 2
a021 Android device info                 a022 Android APDP
a023 Android MSADP                       a024 Android DPO
a025 Android recovery 2                  a026 Android persist
a027 Android modem ST1                   a028 Android modem ST2
a029 Android FSC                         a02a Android FSG 1
a02b Android FSG 2                       a02c Android SSD
a02d Android keystore                    a02e Android encrypt
a02f Android EKSST                       a030 Android RCT
a031 Android spare1                      a032 Android spare2
a033 Android spare3                      a034 Android spare4
a035 Android raw resources               a036 Android boot 2
a037 Android FOTA                        a038 Android system 2
a039 Android cache                       a03a Android user data
a03b LG (Android) advanced flasher       a03c Android PG1FS
a03d Android PG2FS                       a03e Android board info
a03f Android MFG                         a040 Android limits
a200 Atari TOS basic data                a500 FreeBSD disklabel
a501 FreeBSD boot                        a502 FreeBSD swap
a503 FreeBSD UFS                         a504 FreeBSD ZFS
a505 FreeBSD Vinum/RAID                  a580 Midnight BSD data
Press the <Enter> key to see more codes, q to quit:
a581 Midnight BSD boot                   a582 Midnight BSD swap
a583 Midnight BSD UFS                    a584 Midnight BSD ZFS
```

```
a585 Midnight BSD Vinum                  a600 OpenBSD disklabel
a800 Apple UFS                           a901 NetBSD swap
a902 NetBSD FFS                          a903 NetBSD LFS
a904 NetBSD concatenated                 a905 NetBSD encrypted
a906 NetBSD RAID                         ab00 Recovery HD
af00 Apple HFS/HFS+                      af01 Apple RAID
af02 Apple RAID offline                  af03 Apple label
af04 AppleTV recovery                    af05 Apple Core Storage
af06 Apple SoftRAID Status               af07 Apple SoftRAID Scratch
af08 Apple SoftRAID Volume               af09 Apple SoftRAID Cache
af0a Apple APFS                          b300 QNX6 Power-Safe
bb00 Barebox boot loader                 bc00 Acronis Secure Zone
be00 Solaris boot                        bf00 Solaris root
bf01 Solaris /usr & Mac ZFS              bf02 Solaris swap
bf03 Solaris backup                      bf04 Solaris /var
bf05 Solaris /home                       bf06 Solaris alternate sector
bf07 Solaris Reserved 1                  bf08 Solaris Reserved 2
bf09 Solaris Reserved 3                  bf0a Solaris Reserved 4
bf0b Solaris Reserved 5                  c001 HP-UX data
Press the <Enter> key to see more codes, q to quit:
c002 HP-UX service                       e100 ONIE boot
e101 ONIE config                         e900 Veracrypt data
ea00 XBOOTLDR partition                  eb00 Haiku BFS
ed00 Sony system partition               ed01 Lenovo system partition
ef00 EFI system partition                ef01 MBR partition scheme
ef02 BIOS boot partition                 f800 Ceph OSD
f801 Ceph dm-crypt OSD                   f802 Ceph journal
f803 Ceph dm-crypt journal               f804 Ceph disk in creation
f805 Ceph dm-crypt disk in creation      f806 Ceph block
f807 Ceph block DB                       f808 Ceph block write-ahead log
f809 Ceph lockbox for dm-crypt keys      f80a Ceph multipath OSD
f80b Ceph multipath journal              f80c Ceph multipath block 1
f80d Ceph multipath block 2              f80e Ceph multipath block DB
f80f Ceph multipath block write-ahead l  f810 Ceph dm-crypt block
f811 Ceph dm-crypt block DB              f812 Ceph dm-crypt block write-a
f813 Ceph dm-crypt LUKS journal          f814 Ceph dm-crypt LUKS block
f815 Ceph dm-crypt LUKS block DB         f816 Ceph dm-crypt LUKS block wr
f817 Ceph dm-crypt LUKS OSD              fb00 VMWare VMFS
fb01 VMWare reserved                     fc00 VMWare kcore crash protecti
fd00 Linux RAID
```

```
Command (? for help): t
Partition number (1-6): 2
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300): L
Type search string, or <Enter> to show all codes: Linux
8200 Linux swap                          8300 Linux filesystem
8301 Linux reserved                      8302 Linux /home
8303 Linux x86 root (/)                  8304 Linux x86-64 root (/)
8305 Linux ARM64 root (/)                8306 Linux /srv
8307 Linux ARM32 root (/)                8308 Linux dm-crypt
8309 Linux LUKS                          830a Linux IA-64 root (/)
830b Linux x86 root verity               830c Linux x86-64 root verity
830d Linux ARM32 root verity             830e Linux ARM64 root verity
830f Linux IA-64 root verity             8310 Linux /var
8311 Linux /var/tmp                      8312 Linux user\'s home
8313 Linux x86 /usr                      8314 Linux x86-64 /usr
8315 Linux ARM32 /usr                    8316 Linux ARM64 /usr
8317 Linux IA-64 /usr                    8318 Linux x86 /usr verity
8319 Linux x86-64 /usr verity            831a Linux ARM32 /usr verity
831b Linux ARM64 /usr verity             831c Linux IA-64 /usr verity
8500 Container Linux /usr                8501 Container Linux resizable r
8502 Container Linux /OEM customization  8503 Container Linux root on RAI
8e00 Linux LVM                           fd00 Linux RAID
Hex code or GUID (L to show codes, Enter = 8300): 8200
Changed type of partition to 'Linux swap'

Command (? for help): t
Partition number (1-6): 5
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300): EF00
Changed type of partition to 'EFI system partition'

Command (? for help): t
Partition number (1-6): 6
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300): L
Type search string, or <Enter> to show all codes: Linux
8200 Linux swap                          8300 Linux filesystem
8301 Linux reserved                      8302 Linux /home
```

```
8303 Linux x86 root (/)                  8304 Linux x86-64 root (/)
8305 Linux ARM64 root (/)                8306 Linux /srv
8307 Linux ARM32 root (/)                8308 Linux dm-crypt
8309 Linux LUKS                          830a Linux IA-64 root (/)
830b Linux x86 root verity               830c Linux x86-64 root verity
830d Linux ARM32 root verity             830e Linux ARM64 root verity
830f Linux IA-64 root verity             8310 Linux /var
8311 Linux /var/tmp                      8312 Linux user\'s home
8313 Linux x86 /usr                      8314 Linux x86-64 /usr
8315 Linux ARM32 /usr                    8316 Linux ARM64 /usr
8317 Linux IA-64 /usr                    8318 Linux x86 /usr verity
8319 Linux x86-64 /usr verity            831a Linux ARM32 /usr verity
831b Linux ARM64 /usr verity             831c Linux IA-64 /usr verity
8500 Container Linux /usr                8501 Container Linux resizable r
8502 Container Linux /OEM customization  8503 Container Linux root on RAI
8e00 Linux LVM                           fd00 Linux RAID
Hex code or GUID (L to show codes, Enter = 8300): 8e00
Changed type of partition to 'Linux LVM'

Command (? for help):
```

Check all configurations again by entering p. If everything is as expected we can save it by entering w. Again, if this fails, use the partprobe command to save all changes.

bash

```
# FDISK
Command (m for help): p
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Disk model: VMware Virtual S
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: 0x8b8ca071

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sdb1          2048  1026047  1024000  500M 83 Linux
```

```
/dev/sdb2       1026048  2050047  1024000  500M 82 Linux swap / Solaris
/dev/sdb3       2050048  2664447   614400  300M 83 Linux
/dev/sdb4       2664448 16777215 14112768  6.7G  5 Extended
/dev/sdb5       2666496  3383295   716800  350M  c W95 FAT32 (LBA)
/dev/sdb6       3385344  4204543   819200  400M 8e Linux LVM

Filesystem/RAID signature on partition 1 will be wiped.

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

bash

```
# GDISK
Command (? for help): p
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: VMware Virtual S
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 4B226FAA-BFB1-4A6F-9F76-CA70124E6336
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 12578749 sectors (6.0 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         1026047   500.0 MiB   8300  Linux filesystem
   2         1026048         2050047   500.0 MiB   8200  Linux swap
   3         2050048         2664447   300.0 MiB   8300  Linux filesystem
   5         2664448         3381247   350.0 MiB   EF00  EFI system parti
   6         3381248         4200447   400.0 MiB   8E00  Linux LVM

Command (? for help): w

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

```
Do you want to proceed? (Y/N): Y
OK; writing new GUID partition table (GPT) to /dev/sdb.
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.
student@linux-ess:~$
```

We are now able to search for all our partitions in the /proc/partitions folder.

bash

```
student@linux-ess:~$ grep sdb /proc/partitions
   8       16    8388608 sdb
   8       17     512000 sdb1
   8       18     512000 sdb2
   8       19     307200 sdb3
   8       20          1 sdb4
   8       21     358400 sdb5
   8       22     409600 sdb6
```

All partitions are now ready with their different types. We now need to enter the command to install the file systems to these partitions. Again we use the mkfs command for sdb1, sdb3 and sdb5. Sdb2 will be a swap space, we use the command mkswap to accomplish this. For the lvm volume, sdb6, we need to create a physical volume with the pvcreate command.

bash

```
student@linux-ess:~$ sudo mkfs.ext4 /dev/sdb1
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 128000 4k blocks and 128000 inodes
Filesystem UUID: 2860cdfd-1eb5-4fad-b5bf-fcd4b9362009
Superblock backups stored on blocks:
        32768, 98304

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
```

```
Writing superblocks and filesystem accounting information: done

student@linux-ess:~$ sudo mkswap /dev/sdb2
Setting up swapspace version 1, size = 500 MiB (524283904 bytes)
no label, UUID=c7fc81ae-3382-40ab-b37e-7cdeb76535aa
student@linux-ess:~$ sudo mkfs.ext2 /dev/sdb3
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 76800 4k blocks and 76800 inodes
Filesystem UUID: e9e9e303-c9f1-48a6-97d0-d527b0fb7cd0
Superblock backups stored on blocks:
        32768

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

student@linux-ess:~$ sudo mkfs.vfat /dev/sdb5
mkfs.fat 4.2 (2021-01-31)
student@linux-ess:~$ sudo pvcreate /dev/sdb6
  Physical volume "/dev/sdb6" successfully created.
```

All of the partitions are now ready to be mounted, used as swapspace or added to a volume group. We'll check our partitions and mount point with the fdisk and lsblk commands. We can check all physical volumes with the pvs command.

bash

```
student@linux-ess:~$ sudo fdisk -l
...
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Disk model: VMware Virtual S
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: 0x8b8ca071

Device     Boot   Start     End  Sectors  Size Id Type
/dev/sdb1          2048 1026047  1024000  500M 83 Linux
```

```
/dev/sdb2       1026048  2050047  1024000  500M 82 Linux swap / Solaris
/dev/sdb3       2050048  2664447   614400  300M 83 Linux
/dev/sdb4       2664448 16777215 14112768  6.7G  5 Extended
/dev/sdb5       2666496  3383295   716800  350M  c W95 FAT32 (LBA)
/dev/sdb6       3385344  4204543   819200  400M 8e Linux LVM
...
student@linux-ess:~$ sudo lsblk -f /dev/sdb
NAME   FSTYPE      FSVER    LABEL UUID
sdb
├─sdb1 ext4        1.0            2860cdfd-1eb5-4fad-b5bf-fcd4b9362009
├─sdb2 swap        1              c7fc81ae-3382-40ab-b37e-7cdeb76535aa
├─sdb3 ext2        1.0            e9e9e303-c9f1-48a6-97d0-d527b0fb7cd0
├─sdb4
├─sdb5 vfat        FAT16          5555-CA20
└─sdb6 LVM2_member LVM2 001       M5f0lN-AczT-hscz-nzs9-iyVe-f49T-V04MNd
student@linux-ess:~$ sudo pvs
  PV         VG        Fmt  Attr PSize   PFree
  /dev/sda3  ubuntu-vg lvm2 a--   18.22g   8.22g
  /dev/sdb6            lvm2 ---  400.00m 400.00m
```

We can also check all the block devices with partitions with the parted command.

bash

```
student@linux-ess:~$ sudo parted -l
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sda: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  2097kB  1049kB                     bios_grub
 2      2097kB  1904MB  1902MB  ext4
 3      1904MB  21.5GB  19.6GB


Model: VMware, VMware Virtual S (scsi)
```

```
Disk /dev/sdb: 8590MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system     Name                  Flag
 1      1049kB  525MB   524MB   ext4            Linux filesystem
 2      525MB   1050MB  524MB   linux-swap(v1)  Linux swap            swap
 3      1050MB  1364MB  315MB   ext2            Linux filesystem
 5      1364MB  1731MB  367MB                   EFI system partition  boot
 6      1731MB  2151MB  419MB                   Linux LVM             lvm


Model: Linux device-mapper (linear) (dm)
Disk /dev/mapper/myvg0-music: 54.5MB
Sector size (logical/physical): 512B/512B
Partition Table: loop
Disk Flags:

Number  Start  End     Size    File system  Flags
 1      0.00B  54.5MB  54.5MB  ext4


Model: Linux device-mapper (linear) (dm)
Disk /dev/mapper/ubuntu--vg-ubuntu--lv: 10.7GB
Sector size (logical/physical): 512B/512B
Partition Table: loop
Disk Flags:

Number  Start  End     Size    File system  Flags
 1      0.00B  10.7GB  10.7GB  ext4
```

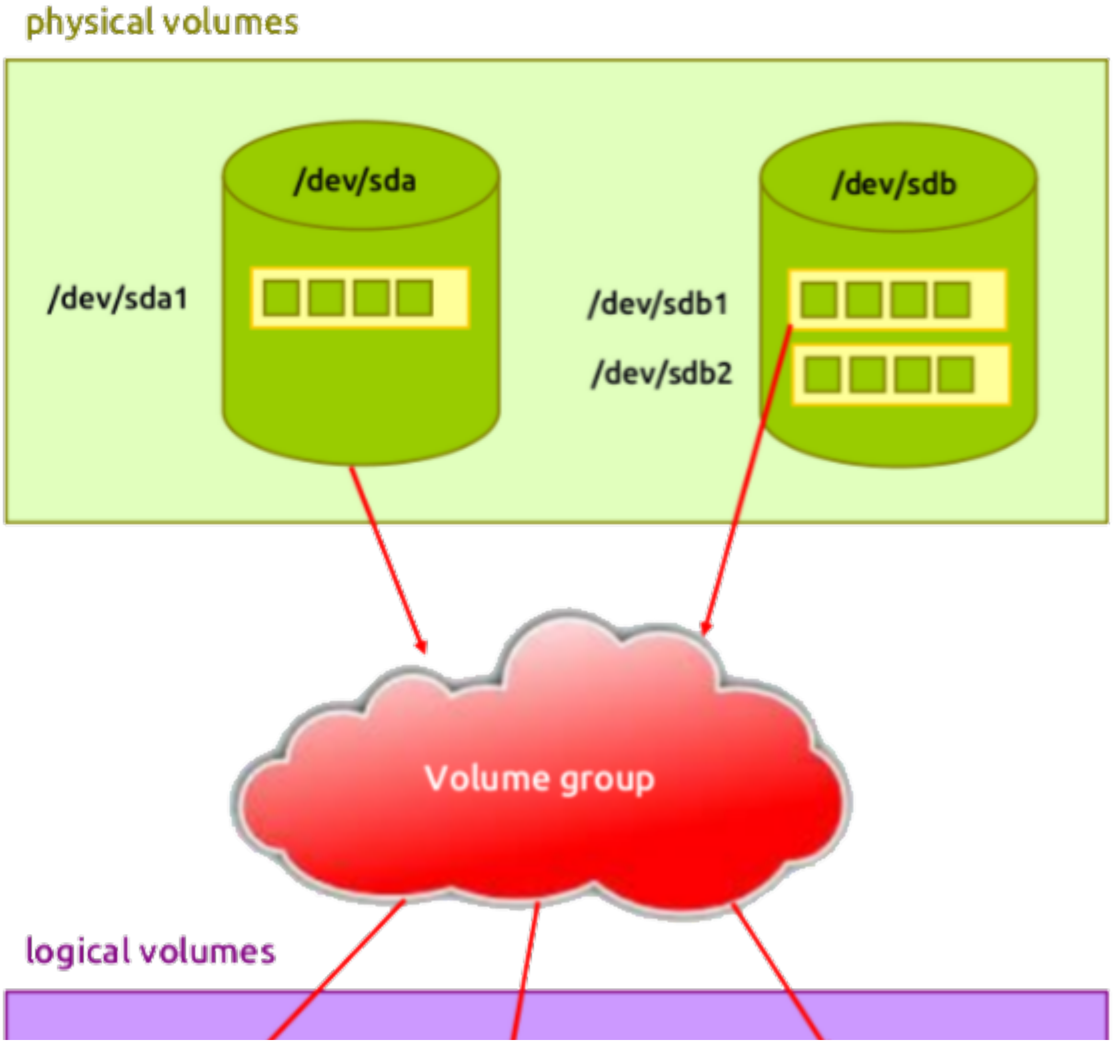
LVMs

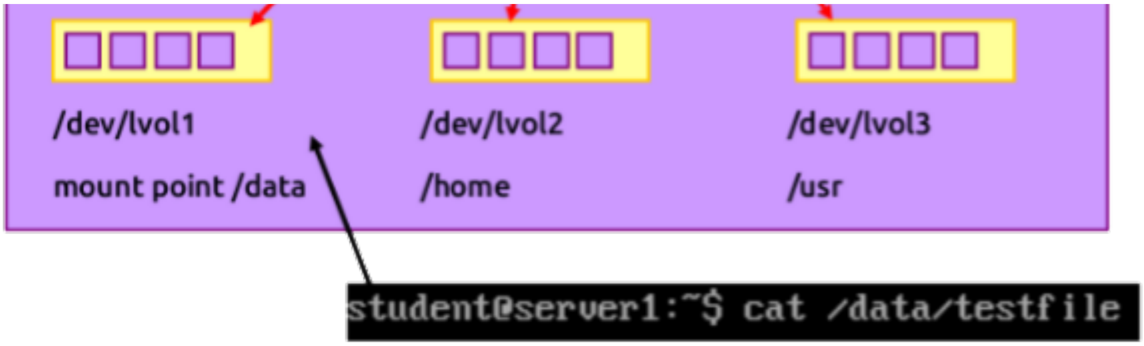
Now that we created a physical volume on our Linux LVM, we'll go a little further into this. Back in the days, when LVM wasn't available, when a drive ran out of free space, we needed to replace it by a larger one and copy everything over

from the old drive. This was time consuming and very inefficient.
LVM gives us more flexibility to add new physical volumes to a volume group. This brings a few advantages:

- Adding drive space is possible even when the logical volume is in use
- You can add physical volumes to a volume group whenever needed
- It’s possible to move data from one physical volume to another, this way we can replace smaller drives with larger ones without any downtime
- Even downsizing a logical volume is possible if its file system supports it.
- LVM also supports advanced possibilities like mirroring and working with clusters.

  LVM OVERVIEW





```
student@server1:~$ cat /data/testfile
```

LVM COMMANDS

Use the command vgs to check for volume groups, pvs to check for physical volumes and lvs to check for logical volumes.

```bash
student@linux-ess:~$ sudo vgs
  VG        #PV #LV #SN Attr   VSize  VFree
  ubuntu-vg   1   1   0 wz--n- 18.22g 8.22g
student@linux-ess:~$ sudo pvs
  PV         VG        Fmt  Attr PSize   PFree
  /dev/sda3  ubuntu-vg lvm2 a--   18.22g   8.22g
  /dev/sdb6            lvm2 ---  400.00m 400.00m
student@linux-ess:~$ sudo lvs
  LV        VG        Attr       LSize  Pool Origin Data%  Meta%  Move Lo
  ubuntu-lv ubuntu-vg -wi-ao---- 10.00g
```

First we'll check our main drive, if we have a Linux LVM partition

```bash
student@linux-ess:~$ sudo fdisk -l /dev/sda
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VMware Virtual S
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: gpt
Disk identifier: A9E92175-1433-43DC-968D-95C5E18A2105

Device       Start      End  Sectors  Size Type
```

```
/dev/sda1     2048     4095     2048    1M BIOS boot
/dev/sda2     4096  3719167  3715072  1.8G Linux filesystem
/dev/sda3  3719168 41940991 38221824 18.2G Linux filesystem
```

With the pvdisplay command we can check if the partition is already in use by a LVM group.

bash

```
student@linux-ess:~$ sudo pvdisplay /dev/sda3
  --- Physical volume ---
  PV Name               /dev/sda3
  VG Name               ubuntu-vg
  PV Size               <18.23 GiB / not usable 3.00 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              4665
  Free PE               2105
  Allocated PE          2560
  PV UUID               9cx3Lm-p95X-Q5fO-ZLSo-KPtf-1ZoN-UyqahD
```

We see that /dev/sda3 has a physical size of 18.23GiB. This volume is added to the volume group: ubuntu-vg. The smallest unit storage that can be granted is 4 MiB.

ℹ EXTRA INFO: 1 MB = 1000000 BYTES = 10^6^ BYTES
1 MIB = 1048576 BYTES = 2^20^ BYTES

To check the information about this volume group use the command vgdisplay.

bash

```
student@linux-ess:~$ sudo vgdisplay
  --- Volume group ---
  VG Name               ubuntu-vg
  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               1
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               18.22 GiB
  PE Size               4.00 MiB
  Total PE              4665
  Alloc PE / Size       2560 / 10.00 GiB
  Free  PE / Size       2105 / 8.22 GiB
  VG UUID               cPT0cP-GoZh-K91H-rSeQ-2ByY-SCtW-dNqE7C
```

With the lvdisplay command, we can check from which volumegroup it is a member, the total size, etc.

bash

```
student@linux-ess:~$ sudo lvdisplay
  --- Logical volume ---
  LV Path                /dev/ubuntu-vg/ubuntu-lv
  LV Name                ubuntu-lv
  VG Name                ubuntu-vg
  LV UUID                o7jRC0-O2XW-z9dW-t2g7-CQ0o-EtCc-VhxL8v
  LV Write Access        read/write
  LV Creation host, time ubuntu-server, 2022-07-09 10:36:52 +0000
  LV Status              available
  # open                 1
  LV Size                10.00 GiB
  Current LE             2560
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0
```

We can see that the logical volume ubuntu-lv is getting allocation space from the volumegroup ubuntu-vg. In the folder /dev/mapper you'll find the file ubuntu--vg-ubuntu--lv, this name refers to its logical volume.

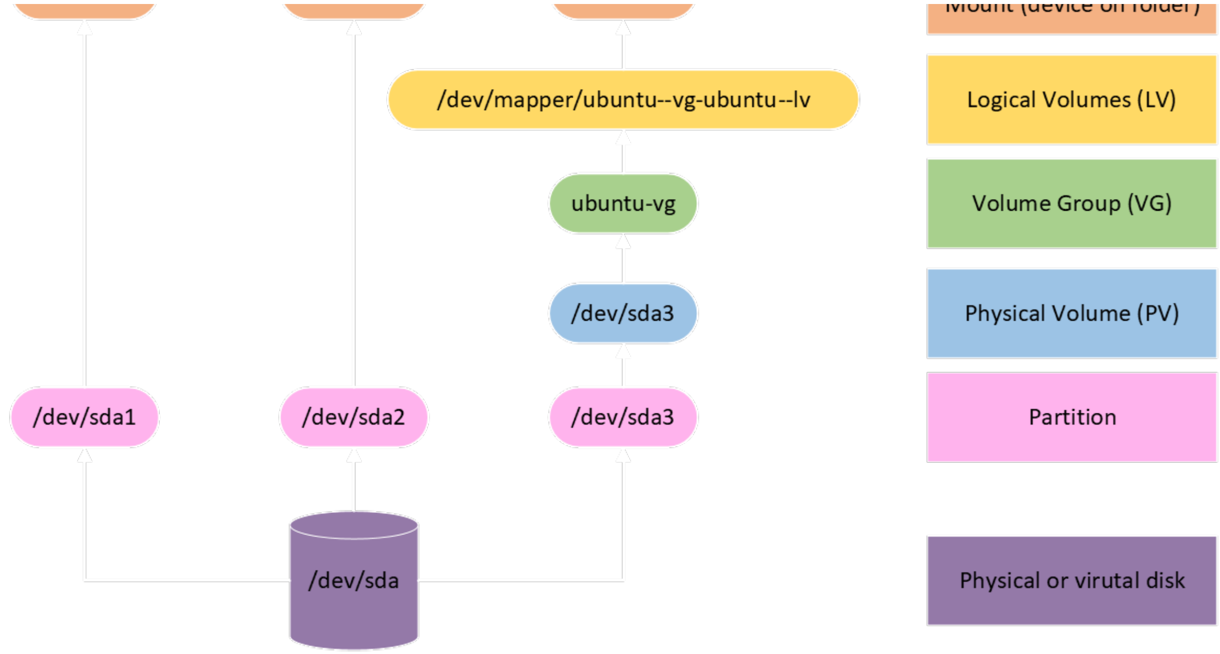
```bash
student@linux-ess:~$ ls /dev/mapper/
control  ubuntu--vg-ubuntu--lv
```

In the fstab file we see that, while booting, the following 2 volumes are automatically mounted with their filesystems. The root- and boot-directory are formatted as ext4. We'll now check the connection between all these groups and volumes.

```bash
student@linux-ess:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name device
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/ubuntu-vg/ubuntu-lv during curtin installation
/dev/disk/by-id/dm-uuid-LVM-0Iw7F1aOuszZbXwuRqub1PoLZIvWxbIGsWuIcTAED0HcQ
# /boot was on /dev/sda2 during curtin installation
/dev/disk/by-uuid/c23d07d6-5019-4858-8c46-4b31cc88e159 /boot ext4 default

student@linux-ess:~$ ls -l /dev/ubuntu-vg/*
lrwxrwxrwx 1 root root 7 Sep 15 12:43 /dev/ubuntu-vg/ubuntu-lv -> ../dm-0
student@linux-ess:~$ ls -l /dev/dm-*
brw-rw----  1 root disk  253, 0 Sep 15 12:43 /dev/dm-0

student@linux-ess:~$ ls -l /dev/mapper/*
crw------- 1 root root 10, 236 Sep  15 21:08 /dev/mapper/control
lrwxrwxrwx 1 root root       7 Sep  15 21:08 /dev/mapper/ubuntu--vg-ubunt
```



As shown in the figure, it starts with one or more physical volumes, from these you can create a volumegroup. From this volumegroup we create logical volumes. We can check these 3 steps with the commands used earlier: pvdisplay, vgdisplay and lvdisplay. We will now create our own volume group and logical volumes from the physical volume created earlier. To create this physical volume we used the pvcreate command, to create a volumegroup we can use a similar command: vgcreate, we name our group myvg0.

bash

```bash
student@linux-ess:~$ sudo vgcreate myvg0 /dev/sdb6
[sudo] password for student:
  Volume group "myvg0" successfully created
student@linux-ess:~$ sudo vgdisplay myvg0
sudo vgdisplay myvg0
  --- Volume group ---
  VG Name               myvg0
  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               396.00 MiB
  PE Size               4.00 MiB
  Total PE              99
  Alloc PE / Size       0 / 0
  Free  PE / Size       99 / 396.00 MiB
  VG UUID               AtpRkW-tbTs-MSIv-mXlY-mgvO-H1bJ-gD7ryc
```

From the free 400MiB, 396 MiB is usable in blocks of 4MiB. We’ll now create a logical volume 'music' with our group with the lvcreate command.

bash

```
student@linux-ess:~$ sudo lvcreate -n music -L 100M myvg0
  Logical volume "music" created.
student@linux-ess:~$ sudo lvs myvg0
  LV    VG    Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%S
  music myvg0 -wi-a----- 100.00m
student@linux-ess:~$ ls /dev/mapper/myvg0*
/dev/mapper/myvg0-music
```

If we want to use this logical volume, we need to give it a file system and mount it to our system. This is possible, as shown before, with the mount command or an entry in the /etc/fstab file.

bash

```
student@linux-ess:~$ sudo mkfs.ext4 /dev/mapper/myvg0-music
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 25600 4k blocks and 25600 inodes

Allocating group tables: done
Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
student@linux-ess:~$ sudo mkdir /mnt/mymusic
student@linux-ess:~$ sudo mount /dev/mapper/myvg0-music /mnt/mymusic/
student@linux-ess:~$ sudo df -h /mnt/mymusic/
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/myvg0-music   90M   24K   83M   1% /mnt/mymusic
student@linux-ess:~$ sudo nano /etc/fstab
/dev/mapper/myvg0-music /mnt/mymusic ext4 default 1 2
```

If we now want to enlarge our logical volume, we can do so without unmounting it. We do need to have free volume in our volumegroup. First you'll need to enlarge the logical volume and afterwards the file system. So first, we'll check the available space in our group and logical volume.

bash

```
student@linux-ess:~$ sudo df -h /mnt/mymusic/
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/myvg0-music   90M   24K   83M   1% /mnt/mymusic
student@linux-ess:~$ sudo vgs myvg0
  VG    #PV #LV #SN Attr   VSize   VFree
  myvg0   1   1   0 wz--n- 396.00m 296.00m
```

We can than extend the volume as possible with the lvextend command.

bash

```
student@linux-ess:~$ sudo lvextend -L +100M /dev/mapper/myvg0-music
  Size of logical volume myvg0/music changed from 100.00 MiB (25 extents)
  Logical volume myvg0/music successfully resized.
```

Afterwards the file system needs to be enlarged as well, this is possible with the resize2fs command for ext filesystems and xfs_growfs for xfs filesystems. To do it in just one step you could have given the -r option within the previous lvextend command.

bash

```
student@linux-ess:~$ sudo resize2fs /dev/mapper/myvg0-music
resize2fs 1.46.5 (30-Dec-2021)
```

```
Filesystem at /dev/mapper/myvg0-music is mounted on /mnt/mymusic; on-line
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/mapper/myvg0-music is now 51200 (4k) blocks long.
```

Check if the file system is enlarged after the configuration.

```bash
student@linux-ess:~$ sudo df -h /mnt/mymusic/
Filesystem              Size  Used Avail Use% Mounted on
/dev/mapper/myvg0-music  184M  120K  175M   1% /mnt/mymusic
```

Now that we have seen how to enlarge a volume, we can try and make it smaller again as well. To start this we need to unmount our volume and check the logical volume for defragmentation with the e2fsck command. Next we need to risize the file system first to the newly wanted smaller size. Afterwards we can make the volumegroup smaller. Now we can remount the logical volume and check if all configurations were successful.

```bash
student@linux-ess:~$ student@linux-ess:~$ sudo umount /mnt/mymusic
student@linux-ess:~$ sudo e2fsck -f /dev/mapper/myvg0-music
e2fsck 1.46.5 (30-Dec-2021)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/mapper/myvg0-music: 11/51200 files (9.1% non-contiguous), 4262/51200
student@linux-ess:~$ sudo resize2fs /dev/mapper/myvg0-music 50M
resize2fs 1.46.5 (30-Dec-2021)
Resizing the filesystem on /dev/mapper/myvg0-music to 12800 (4k) blocks.
The filesystem on /dev/mapper/myvg0-music is now 12800 (4k) blocks long.

student@linux-ess:~$ sudo lvreduce -L 50M -r /dev/mapper/myvg0-music
  Rounding size to boundary between physical extents: 52.00 MiB.
fsck from util-linux 2.37.2
/dev/mapper/myvg0-music: clean, 11/25600 files, 2646/12800 blocks
```

```
resize2fs 1.46.5 (30-Dec-2021)
Resizing the filesystem on /dev/mapper/myvg0-music to 13312 (4k) blocks.
The filesystem on /dev/mapper/myvg0-music is now 13312 (4k) blocks long.

  Size of logical volume myvg0/music changed from 200.00 MiB (50 extents)
  Logical volume myvg0/music successfully resized.
student@linux-ess:~$ sudo mount /dev/mapper/myvg0-music /mnt/mymusic/
student@linux-ess:~$ sudo df -h /mnt/mymusic/
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/myvg0-music   42M   72K   39M   1% /mnt/mymusic
```

The mount command

We’ve now mostly mounted our volumes and drives with the mount command and shown off the /etc/fstab file. But the mount command isn’t only used to mount local storage devices. We can also mount network directories with NFS or Samba, mount image files and mount drives or USB flash drives which didn’t automount. To check what file systems are supported by the kernel to mount, check the /proc/filesystems file. These file systems are supported by the kernel, but not necessarily by your Linux distribution!

bash

```
student@linux-ess:~$ cat /proc/filesystems
nodev   sysfs
nodev   tmpfs
nodev   bdev
nodev   proc
nodev   cgroup
nodev   cgroup2
nodev   cpuset
nodev   devtmpfs
nodev   configfs
nodev   debugfs
nodev   tracefs
nodev   securityfs
nodev   sockfs
nodev   bpf
nodev   pipefs
nodev   ramfs
nodev   hugetlbfs
nodev   devpts
        ext3
        ext2
        ext4
        squashfs
        vfat
nodev   ecryptfs
        fuseblk
nodev   fuse
nodev   fusectl
nodev   mqueue
nodev   pstore
        btrfs
nodev   autofs
```

To check all modules which can be loaded by your kernel when a file system is mounted use the command:

bash

```
student@linux-ess:~$ ls /lib/modules/$(uname -r)/kernel/fs
9p      befs            ceph    efs    freevxfs  hfsplus  ksmbd  nfs_comm
adfs    bfs             cifs    erofs  fscache   hpfs     lockd  nfsd
affs    binfmt_misc.ko  coda    exfat  fuse      isofs    minix  nilfs2
afs     btrfs           cramfs  f2fs   gfs2      jffs2    netfs  nls
autofs  cachefiles      dlm     fat    hfs       jfs      nfs    ntfs
```

For more information check the manpage on filesystems: man fs

Swapspace

Up next, we check how to use a swapspace. We already created one before, but did not add it to our usable swapspace. The swapspace is used when your system runs out of RAM and needs to offload any unused data, which is needed later on again. To create a swapspace we use the command mkswap, to turn the swapspace on or off, use the swapon or swapoff command. We'll recreate adding swapspace by the following example. First we check the available space at this time.

```bash
student@linux-ess:~$ free -h
               total        used        free      shared  buff/cache   av
Mem:           3.8Gi       304Mi       2.8Gi       1.0Mi       678Mi
Swap:             0B          0B          0B
```

We'll now create a file of 1 GiB and make a swapspace out of this. At the end we turn it on to make it usable for our system. We can check this by just entering the swapon command without parameters. We can also check the full available swapspace as shown earlier.

```bash
student@linux-ess:~$ sudo dd if=/dev/zero of=/var/opt/myswap bs=1M count=
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 4.21084 s, 255 MB/s
student@linux-ess:~$ sudo chmod 0600 /var/opt/myswap
student@linux-ess:~$ sudo mkswap /var/opt/myswap
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=0a79aa31-fca6-4ad7-893b-4d3da75233fb
student@linux-ess:~$ sudo swapon /var/opt/myswap
student@linux-ess:~$ swapon
NAME            TYPE  SIZE USED PRIO
/var/opt/myswap file 1024M   0B   -2
student@linux-ess:~$ free -h
               total        used        free      shared  buff/cache   av
Mem:           3.8Gi       308Mi       1.8Gi       1.0Mi       1.7Gi
Swap:          1.0Gi          0B       1.0Gi
```

Again we can add this to the /etc/fstab file to make this extra swapspace permanently available.

```bash
student@linux-ess:~$ sudo nano /etc/fstab
/var/opt/myswap swap swap defaults 0 0
```

We use swap in the second field as there is no mounting point for swapspace. To turn on any swapspace that is mentioned in the /etc/fstab file, for example with the option noauto, immediately we can use the swapon -a command. If we want to remove swapspace we fist need to check that it isn’t in use. If this is the case we can unmount it by the swapoff command.

```bash
student@linux-ess:~$ swapon
NAME            TYPE  SIZE USED PRIO
/var/opt/myswap file 1024M   0B   -2
student@linux-ess:~$ sudo swapoff /var/opt/myswap
```

‹ Previous
11 Managing running processes

Next ›
Lab