Storage administration:
Topics to be covered:

-Disk Management:
-Partitioning
-File system
-Mounting
-fstab file
-Swap management:
-LVM
-Extending/Reduce LVM
-LVM Snapshot

partition is a fix amount of disk space with start and end from the total HD that will be mounted to Directory to be accessed. To see this space we can use the following command

```
[root@server ~]#df -h
[root@server ~]#lsblk command.
```

Before we decide to make disk partitioning. We need to know the business needs and what will this partition used?
user data, or it is business application that needed to grow and that is the growth rate of the data or may be used with database applications. So we can decide how much space that we need to allocate?

Note: there is a common problem that making the partitions with the same size, can lead to miss utilization of disk space.

Disk partitioning:

1. We need to decide either we use MBR or GPT system (note) if you changed from GPT to MBR or Vise versa your data will be lost.
2. Use one of the following tools fdisk (MBR System) or gdisk (GPT) or parted or gparted.
3. To see your disk and partitions via kernel we use the following command
   ```
   [root@server ~]# cat /proc/partitions
   ```
4. We are going to use fdisk tool so to list your disk we used

Note: if you added disk online it will not be seen via kernel until you reboot.

```
[root@server ~]#fdisk -l /dev/sda
```

To show the menu of command options:
For help commands ==> m
Delete a partition ==> d
Lists known partition types ==> l
Adds a new partition ==> n
Prints the partition table ==> p
Quits without saving changes ==> q
Writes table to disk and exit ==> w

Note: if the partition is not available in /proc/partitions, so you have to use partprobe or reboot.

[root@server ~]#partprobe /dev/partition_name   ==> to let the kernel reread the disk if the partition table changed.

After creating partition we have to create on it file system (FS) using the command.

```
[root@server ~]#mkfs.fs_name /dev/partion_name or
[root@server ~]#mkfs -t fstype
-b block size
-I no. of inodes  /dev/partiton_name
-f force to change the previous file system
-i size
```

To check the file system and its feature dumpe2fs /dev/sdb1 (works only with ext2/ext3/ext4 file system)

Note: for each file system have special use with mkfs, dumpe2fs, fsck, badblocks, e2fsck commands.

Each file system has its own purpose and features that it support (max file size, support quota, ACL, Compression, Archiving, journaling and snapshot).

In RHEL 6 we used ext4 as default file system, but in RHEL 7 we used xfs as default file system. In the next versions of RHEL7 they used brtfs. There are a lot of FS used in linux like GFS (Active/Active) Cluster, vFAT (windows) and Gluster FS (DFS).

Link for well known file systems and its features:
http://www.electronicdesign.com/industrial/what-s-difference-between-linux-ext-xfs-and-btrfs-file-systems

After creating the FS we need to mount the device to the system to be accessed, so we have to use mount point.

We have to create directory to be mounted and have mount point for example /dev/sda1 ====> /test.

```
[root@server ~]#mount /dev/partiton_name  /test
```

Note: we can use a lot of options -o like remount, rw, ro, quote,acl in mount command.

Mount only used to mount what is in /etc/fstab file.

```
[root@server ~]#mount | grep /dev ==> to see the partitions
```

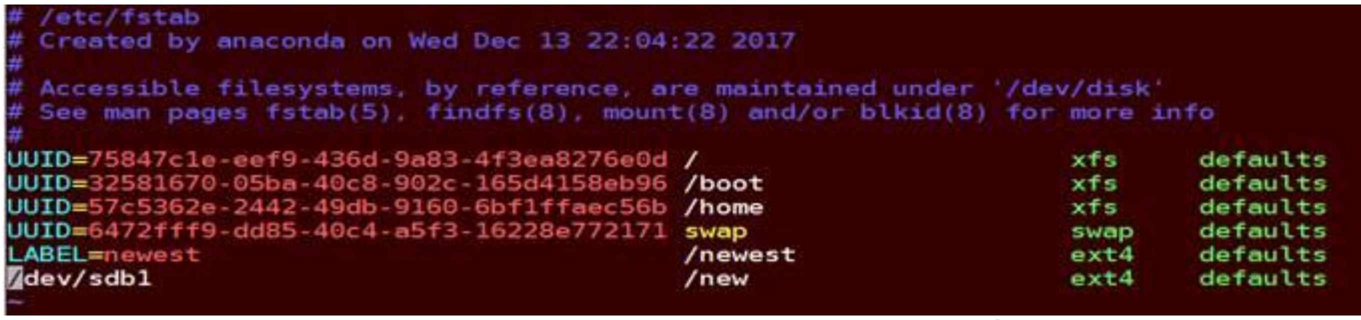
```
[root@server ~]#umount /test ==> To disconnect the device from the mount point.
[root@server ~]#umount -f  /dev/partiton_name
```

If we restarted our machine we will lost our mount point because it is not saved in a file, so we need to save our mount point to be accessed permanently.

We will use /etc/fstab to set our mount point:

fstab file, This file is very critical file if something wrong with this file we cannot boot our system normally:

```
DEVICE   MOUNT_POINT   FILE_SYSTEM_USED
FS_MOUNTING_OPTIONS   DUMP  FSCK
```


```
# /etc/fstab
# Created by anaconda on Wed Dec 13 22:04:22 2017
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=75847c1e-eef9-436d-9a83-4f3ea8276e0d /                       xfs     defaults
UUID=32581670-05ba-40c8-902c-165d4158eb96 /boot                   xfs     defaults
UUID=57c5362e-2442-49db-9160-6bf1ffaec56b /home                   xfs     defaults
UUID=6472fff9-dd85-40c4-a5f3-16228e772171 swap                    swap    defaults
LABEL=newest                              /newest                 ext4    defaults
/dev/sdb1                                 /new                    ext4    defaults
~
```

Mounting methods:

1. Device mounting:
   [root@server ~]#mount /dev/sdb1 /data ==> to make temporary mount.
   [root@server ~]#vim /etc/fstab ==> to make it permanent mount.

2. Label mounting:
   [root@server ~]#e2label /dev/sdb2 newest ==> we named our device by label
   [root@server ~]#blkid ==> to check our label
   [root@server ~]#mount LABEL=newest /newest ==> to make temporary mount.
   [root@server ~]#vim /etc/fstab ==> to make it permanent mount.

3. UUID Mounting:

   We can generate for the device UUID and it is unique for each device on the system via command uuitgen /dev/sdb1 and mount the device with its UUID in /etc/fstab.

Note: After editing in the /etc/fstab we have to test our file by using mount command (mount -a) if it fail so there is something wrong in /etc/fstab ,but if it is success so no action on screen.

Backup and FS Check:
DUMP 0 means no dump needed if 1 so we can use dump command.
FSCK 0 no fsck on partition, 1 used for the fsck on rootfs, 2 for the other partitions to be checked after the rootfs.

Note: there is tools that can we use for xfs files system like xfs_admin -L labelname /dev/sdb1 and mount -L labelname /newest.

Swap:

Swap space allows processes to use more memory than actually exists on the system. If the amount of memory requested by the process running on the system exceeds the amount of available RAM, the Linux kernel can swap some of the pages of memory being used by sleeping or idle processes to disk to make room for the additional memory needed by running processes or new processes.

To display your current swap area

```
[root@server ~]#swapon -s or free -h
```

Add new swap partition:

Using the fdisk utility to create a partition, Set the system ID to the value hex 82:. Use the t command within fdisk to change a partition's system ID to Linux swap and save the changes. Use partprobe to force the system to recognize the changes.

```
[root@server ~]#cat /proc/partitions  --> show partitions
[root@server ~]#fdisk /dev/sdb
```

Format swap partition:

```
[root@server ~]## mkswap /dev/sdc2
```

To use it as a swap space

```
[root@server ~]#swapon /dev/sdc2
```

To make it permanent

```
[root@server ~]#vim /etc/fstab --> /dev/sdc2 swap swap defaults 0 0
```

Add temporary file as a swap:

Create a local file of the required size using the dd command.
Format this file just as if it were a partition device file.

```
dd if=/dev/zero of=/myswap bs=1024 count=1024
1024+0 records in
1024+0 records out
```

```
[root@server ~]# ls -l /myswap -rw-r--r-- 1 root root 1048576 Jul 6 06:37 /myswap
```

We must change the swap permissions to be accessible via root only

```
[root@server ~]# chmod 600 /myswap
[root@server ~]#mkswap /myswap Setting up swapspace version 1, size = 1044 kB
```

Turn on the swap to the system:

```
[root@server ~]#swapon /myswap
[root@server ~]#swapon –s
Filename     Type          Size        Used      Priority
/dev/hda2  partition   327672   2868          -1
/myswap     file           1016         0            -2
```

To make the swap perment:

```
[root@server ~]# vim /etc/fstab
/myswap           swap              swap          defaults           0    0
```

To remove file from swap area

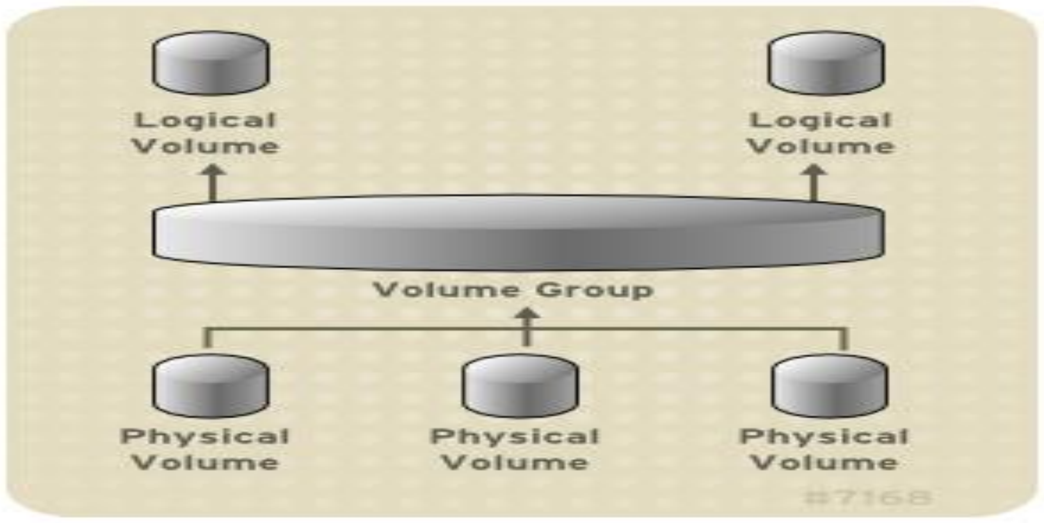
```
[root@server ~]#swapoff /myswap
```

Logical Volume Manager (LVM):
If one of our partitions is now utilized with around 95 % and files is continue to grow, so we are in a very big problem but with LVM everything is file:

LVM provides:

1. Online Growth of File System without Data loss.
2. One or more physical volume (hard disk/ partition/RAID array, or SAN device) are assigned to a volume group (VG).
3. Can be used to make multiple volumes.
4. Volume can span multiple disks.
5. Advanced option such as snapshot which allows you to make backup of files even if they are open.
6. Support 15 partition and 256 logical volume.
7. If you have failing disk, you can easily replace it with new disk.
8. We can reduce the size of the volume.

LVM Make abstraction of the storage, first we create physical volumes (PV) from this physical volumes we create a very large Storage pool, and from this pool we can assign multiple numbers of logical volumes.



From the physical volume we have physical extents (PE), it is blocks of the data of the storage that sometimes called chunks, default PE 4MB and can be ranged from 1 MB to 64 MB.

LVM Implementation:

1. Create disks or partitions using fdisk
2. Change the partition tag to Linux lvm code 8e
3. Create PV
4. Test it with pvs or pvdisplay
5. Create VG give it name and test it using vgs or vgdisplay
6. Create the Logical volume and test it by lvs or lvdispaly

Create physical volumes:

fdisk /dev/sdb ==> create partition of type 0x8e

Then we have to initialize physical volume

```
[root@server ~]#pvcreate /dev/sdb3
[root@server ~]#pvcreate /dev/sdb4
[root@server ~]#pvdisplay  ==> To show physical volume
[root@server ~]#pvs ==> basic info
[root@server ~]#pvremove name ==> To remove physical volume
```

Create volume groups which function equivalent to a disk:

[root@server ~]#vgcreate vg0 /dev/sdb3 /dev/sdb4 ==> this will create device file in /dev/vg0

NOTE: we can create vg without making pvcreate to the partitions, it will make it automatic

```
[root@server ~]#vgdisplay [name] ==> to show the volume group
[root@server ~]#vgs [name] ==> basic info
[root@server ~]#vgremove name  ==> to remove volume group
```

Create logical volumes which is equivalent to a partition:

```
[root@server ~]#lvcreate –L 2G -n lv0 vg0
```

-L à size
-n àname

```
[root@server ~]#lvdisplay [name] ==> to show logical volume
[root@server ~]#lvs [name] ==> basic info
```

Lvcreate options:

-l ==> no of extends
-l 100%FREE ==> to take all free space.
-l 30%vg ==> to take 30% of the volume group

Note: both devices are pointing to /dev/dm-* where dm is the device mapper and * means no. of devices based on the Virtual file System (VFS).
It will create device file in /dev/vg0/lv0 or /dev/mapper/vg0-lv0, now we can use /dev/vg0/lv0 as block device.

VFS Device:
-RAID
-LVM
-LUKs
-Multipathing

```
[root@server ~]#lvremove name ==> to remove the logical volume.
```

Note: we can partition the logical volume and we can create the hard disk devices into the volume group.

Create file system

```
[root@server ~]# mkfs -t ext4 /dev/vg0/lv0
```

Mount the file system

```
[root@server ~]#mkdir /lvm-ext4
[root@server ~]#mount /dev/vg0/lv0 /lvm-ext4
```

Note: To add to permeant mount, we must edit in /etc/fstab file.

Extend/Reduce a volume group:

Resize a file system sounds easy, but you need to be very aware of what you are doing otherwise you will end up with big mess and it will be very difficult to fix.

To extend the file system, you can't begin with the file system itself, you need available disk space on LV, and to have that you need possibly space in VG as well.

To make file system bigger, you need to start all the way from PV
1- You need to create new PV
2- You need to assign the new PV to the VG
3- You can grow the LV
4- You can extend the FS as well

Note: whenever you change an active PV, LV, or VG, unmount the volume first.
Note: lvextend don't extend a file system in LV automatically

```
[root@server ~]#fdisk /dev/sdc
[root@server ~]#pvcreate /dev/sdc1
[root@server ~]#vgextend vg0 /dev/sdc1
[root@server ~]#lvextend -L +1G /dev/vg0/lv1
[root@server ~]#xfs_growfs  /dev/vg0/lv1   (for XFS File System only)
[root@server ~]#resize2fs  /dev/vg0/lv1  (For Other File systems)
```

Or

```
[root@server ~]#lvextend -r -L +1G  /dev/vg0/lv1 ==> to have one step
```

Or

```
[root@server ~]#lvextend -l +100%FREE -r /dev/vg-1/lv-1
```

Options can be used with lvextend:

```
[root@server ~]#lvextend -L +1G /dev/vg0/lv1
```

-i ==> to specify a size in PE
-l +100%free ==> to ensure that all available disk space in VG is added
-l 128 ==> resize the logical volume to exactly 128 extent in size
-l +128 ==> add 128 extent to the current size
-L 128M ==> resize the logical volume to exactly 128 M
-L +128M ==> add 128 M to the current size of the logical volume
-r ==> is no matter what the file system you use, it resize the file system as well

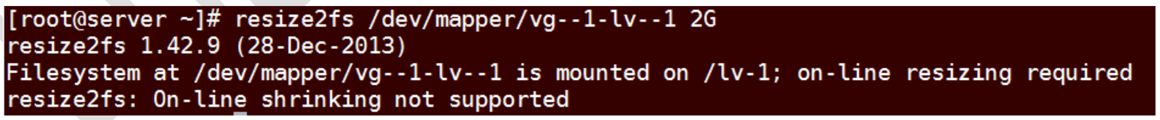
Also we have to reduce operation, which mean we need to make file system smaller, can you imagine what happen if you take the wrong order here.

For example start with reduce the logical volume without reducing the file system first, it will mean the fs will be bigger than the logical volume that is below it and fsck will see the problem and going probably to mess up all your data.

So if you are going to reduce
1- Reduce the file system
2- Reduce the size of LV
3- Reduce the size of VG
4- Take away the PV you don't need anymore

Note: xfs file system did not support reduction of the file system
Note: to reduce the size of the file system it must be unmounted first

If the files system is ext2/ext3/ext4 we use the tool resize2fs
[root@server ~]#umount /lv-1


```
[root@server ~]# resize2fs /dev/mapper/vg--1-lv--1 2G
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/mapper/vg--1-lv--1 is mounted on /lv-1; on-line resizing required
resize2fs: On-line shrinking not supported
```

```
[root@server ~]# resize2fs /dev/mapper/vg--1-lv--1 2G
[root@server ~]# e2fsck -f /dev/mapper/vg--1-lv--1
[root@server ~]# resize2fs /dev/mapper/vg--1-lv--1 2G
[root@server ~]#lvreduce -L 2097152k /dev/vg-1/lv-1
[root@server ~]#vgreduce vg0 /dev/sda3
```
OR

```
[root@server ~]#umount /lv-1
[root@server ~]# lvreduce -L 2G -r /dev/vg-1/lv-1
```

Note: it can cause an error coz /dev/sda3 have data, So we need to move data from physical volume to another volume.

```
[root@server ~]#pvmove /dev/sda3 /dev/sdb4
```

To make reduction of XFS File system but data might be lost :
To reduce the xfs file system we have first take backup from data and umount the device then resize the file system and reduce the logical volume.

```
[root@server ~]# umount /mnt/
[root@server ~]# fsadm -l resize /dev/VG0/new-data
[root@server ~]# lvreduce -L 6G /dev/VG0/new-data
[root@server ~]# lvs
```

===

Tasks:

1. Use fdisk -l to locate information about the partition sizes.
2. Use fdisk to add a new logical partition that is 2GB in size.
3. Did the kernel feel the changes? Display the content of /proc/partitions file? What did you notice? How to overcome that?
4. Make a new ext4 file system on the new logical partition you just created.
   Bonus: Try creating the ext4 filesystem with 2k blocks and one inode per every 4k (two blocks) of filesystem.

5. Create a directory, name it /data.
6. Add a label to the new filesystem, name it data.
7. Add a new entry to /etc/fstab for the new filesystem using the label you just create.
8. Mount the new filesystem
9. Display your swap size.
10. Create a swap file of size 512MB.
11. Add the swap file to the virtual memory of the system.
12. Display the swap size.
13. Use the fdisk command to create 2 Linux LVM (0x8e) partitions using "unpartitioned" space on your hard disk. These partitions should all be the same size; to speed up the lab, do not make them larger than 300 MB each.

Make sure to write the changes to disk by using the w command to exit the fdisk utility. Run the partprobe command after exiting the fdisk utility.

14. Initialize your Linux LVM partitions as physical volumes with the pvcreate command. You can use the pvdisplay command to verify that the partitions have been initialized as physical volumes.
15. Using only one of your physical volumes, create a volume group called test0. Use the vgdisplay command to verify that the volume group was created.
16. Create a small logical volume (LV) called data that uses about 30 percent of the available space of the test0 volume group. Look for VG Size and Free PE/Size in the output of the vgdisplay command to assist you with this. Use the lvdisplay command to verify your work.
17. Create an xfs filesystem on your new LV.
18. Make a new directory called /data and then mount the new LV under the /data directory. Create a "large file" in this volume.
19. Enlarge the LV that you created in Sequence 1 (/dev/test0/data) by using approximately 25 percent of the remaining free space in the test0 volume group. Then, enlarge the filesystem of the LV.
20. Verify that the file /data/bigfile still exists in the LV. Run the df command and check to verify that more free disk space is now available on the LV.
21. Use the remaining extents in the test0 volume group to create a second LV called docs.
22. Run the vgdisplay command to verify that there are no free extents left in the test0 volume group.
23. Create an xfs filesystem on the new LV, make a mount point called /docs and mount the docs LV using this mount point.
24. Add all of the remaining unused physical volumes that you created in Sequence 1 to the test0 volume group.
25. If you run vgdisplay again, there now should be free extents (provided by the new physical volumes) in the test0 volume group. Extend the docs LV and underlying filesystem to make use of all of the free extents of the test0 volume group. Verify your actions.