

Lesson 3

Linux File systems and Network File Service (NFS)

Contents

1.	Setting up an XFS filesystem based on LVM.....	2
2.	Filesystem Access Control Lists.....	14
3.	Exporting directories on the NFS Server - for remote files access	18
4.	Mounting exported directories on the NFS Client	21
5.	NFS Export options	23
6.	Linux File Permissions vs NFS Permissions	26
7.	Accessing exported directories by mounting pseudo-root	28
8.	Mount NFS exported directory on bootup with /etc/fstab.....	31
9.	Enable a Shared Folder (Host) for a Virtual Machine (Linux)	32

1. Setting up an XFS filesystem based on LVM

Linux System and user files are stored in a filesystem. The filesystem provides all the required storage and access control management features. Over the years, there have been many different types of Linux filesystem have been evolved and many of them are still in use.

Oracle Linux 8 ,9 are using XFS as its default filesystem.

Linux Volume Manager (LVM) is a commonly used utility which can help to map multiple disk partitions into one single disk volume. A Linux filesystem can then build on to it. The LVM allows cross partitions and cross disks expansion seamlessly.

In other words, we can create a single XFS filesystem cross multiple physical disks to form on a huge logical disk storage that managed by LVM.

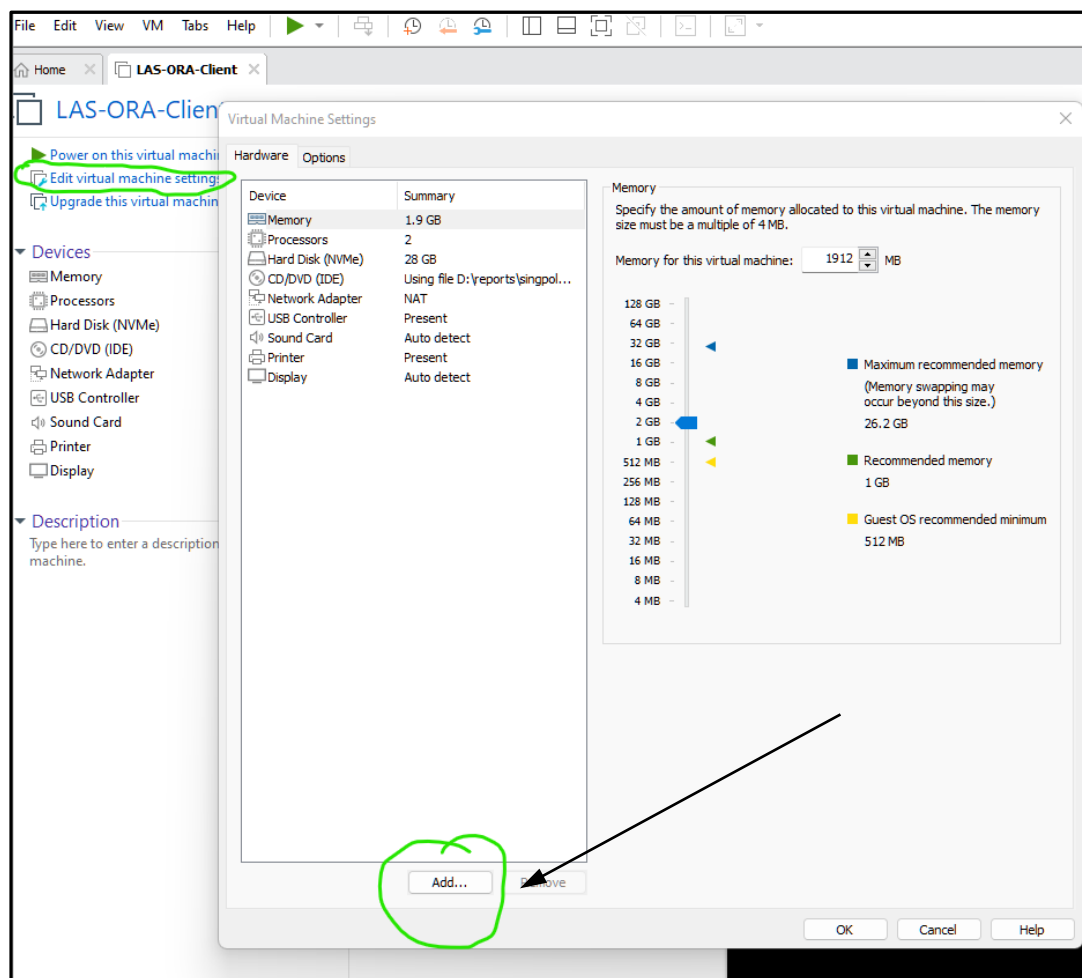
We will be creating an additional VMware virtual disk to our client VM and manage it with LVM. We will format the volume with the XFS filesystem.

(Ref: <https://www.linuxtechi.com/create-extend-xfs-filesystem-on-lvm/>)

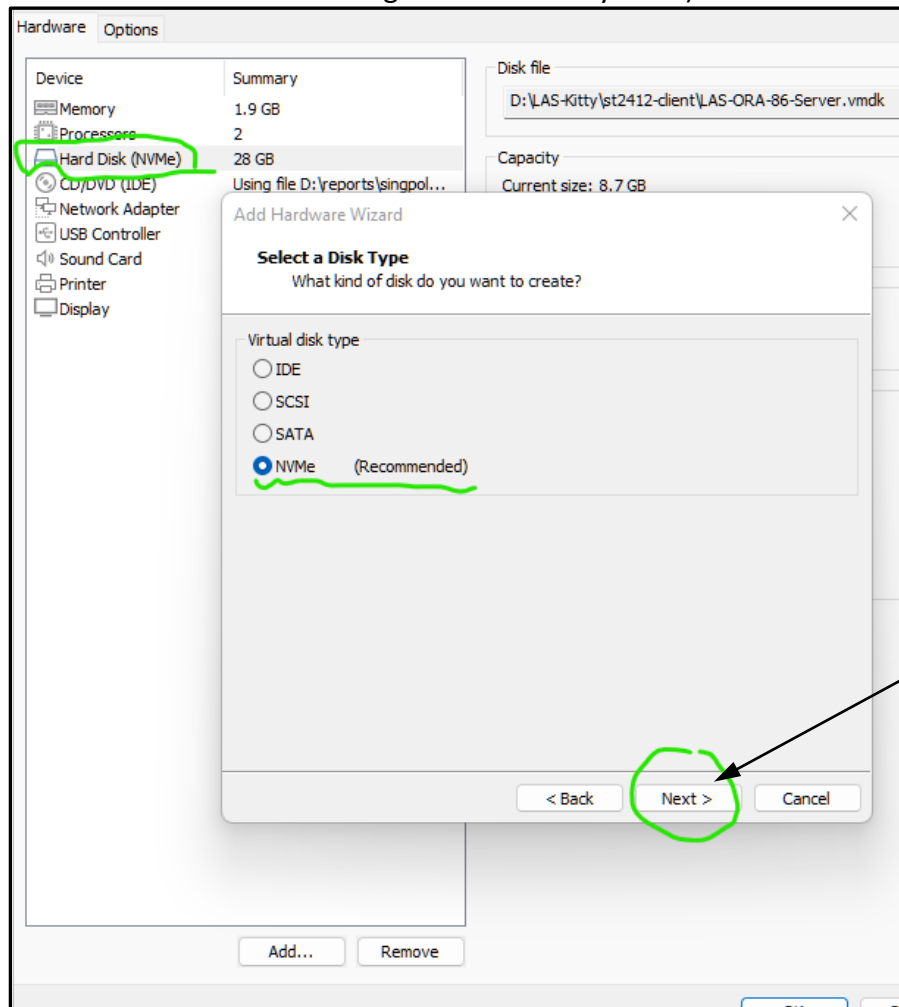
On client (Ensure it is powered off):

Creating a new VM disk

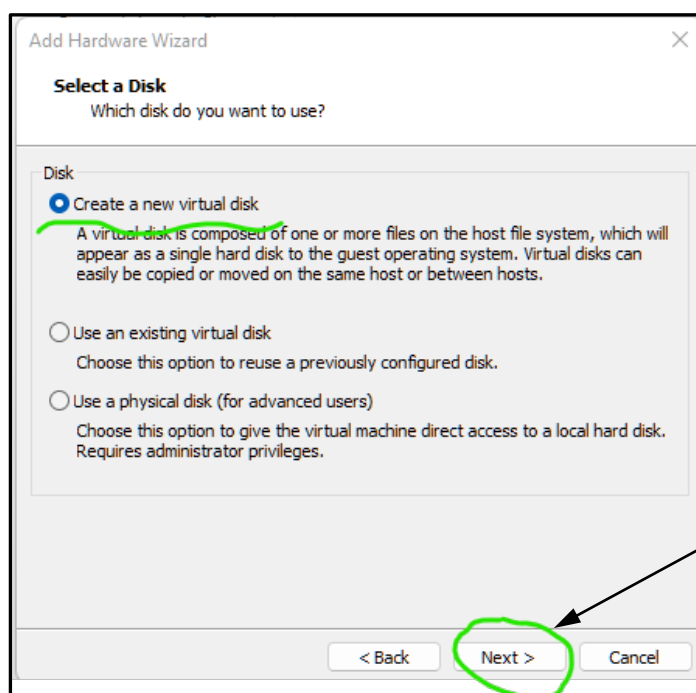
1. Open the VM setting menu and click on the Add button. In the pop up screen, select the **Hard Disk** in the Hardware type and click the next button.



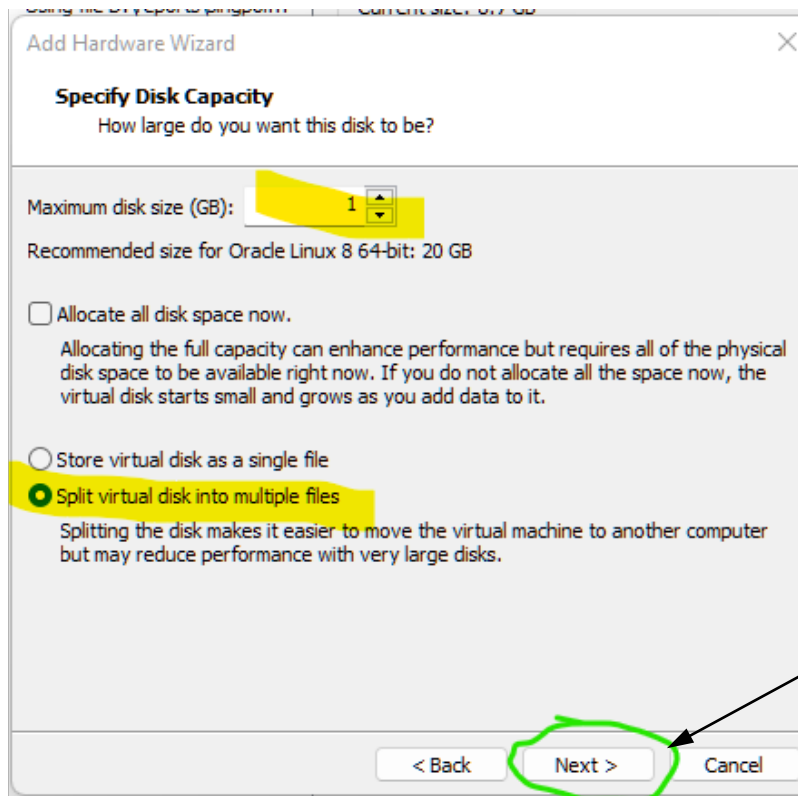
2. In the next screen accept the **recommended** disk type (can be SCSI or NVMe - depending on the actual hardware configuration of the system.) click the Next button.



3. In the next screen accept the default (Create a new virtual disk) option and click the Next button.



4. In the next screen, do not take the default (usually is 20Gb) just specify **1 GB** as the disk capacity, and click the Next button. (You may enter **0.5** to get a smaller disk if needed.)



5. Accept the default filename in the next screen and Click **Finish** to complete the new virtual disk creation.
Now you may start your client VM.

On client (after it is powered on)

1. Login as root (or use sudo if your are logged in as student) to use the lshw (list hardware) command to find out the logical device name of all the disks. Type

```
lshw -class disk
```

You should be able to see the list contains a cdrom, a 30Gb harddisk (or NVME disk), and a 1Gb disk (The one that you have just allocated.)

For the following sample listing, the 1 GB new disk is mapped to /dev/nvme0n2.
(You should get a different logical disk name.)

```
[student@client ~]$ sudo lshw -class disk
*-cdrom
  description: DVD-RAM writer
  product: VMware IDE CDR10
  vendor: NECVMWar
  physical id: 0.0.0
  bus info: scsi@1:0.0.0
  logical name: /dev/cdrom
  logical name: /dev/sr0
  version: 1.00
  capabilities: removable audio cd-r cd-rw dvd dvd-r dvd-ram
  configuration: ansiversion=5 status=open
*-namespace:0
  description: NVMe disk
  physical id: 1
  bus info: nvme@0:1
  logical name: /dev/nvme0n1
  size: 28GiB (30GB)
  capabilities: partitioned partitioned:dos
  configuration: logicalsectorsize=512 sectorsize=512 signature=b9000f7d wwid=nvme.15ad-564d57617265204e564d455f30303030-564d77617265205669727475616c204e564d65204469736b-00000001
*-namespace:1
  description: NVMe disk
  physical id: 2
  bus info: nvme@0:2
  logical name: /dev/nvme0n2
  size: 1GiB (1073MB)
  configuration: logicalsectorsize=512 sectorsize=512 wwid=nvme.15ad-564d57617265204e564d455f30303030-564d77617265205669727475616c204e564d65204469736b-00000002
[student@client ~]$
```

2. While `lshw` can provide the hardware information of the disks of the system. You may also try the other command, `lsblk`, which is a little more sophisticated as it lists all block devices (ie. disks) with the partitions information.

Type "`lsblk`".

```
[student@client ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0          11:0    1 1024M  0 rom
nvme0n1      259:0    0   28G  0 disk
├─nvme0n1p1  259:1    0    1G  0 part /boot
├─nvme0n1p2  259:2    0   27G  0 part
│   └─ol-root 252:0    0 24.2G  0 lvm  /
│       └─ol-swap 252:1    0  2.8G  0 lvm  [SWAP]
└─nvme0n2    259:3    0    1G  0 disk
```

Take note that `lsblk` does not require root to run. The highlighted device, `nvme0n2` (We just created in the previous steps), has no partition defined at this point. Furthermore, the device names shown were incomplete as they have omitted the leading `/dev/` portion.

Viewing and creating new partitions onto the new disk

3. Type "`fdisk -l <logical disk device name>`" to verify the logical disk device name is correct and view the partition information of the targeted disk. For the newly added disk (`/dev/nvme0n2`), it **may not** have any existing partitions as it is brand new.

```
[student@client ~]$ sudo fdisk -l /dev/nvme0n2
[sudo] password for student:
Disk /dev/nvme0n2: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[student@client ~]$
```

4. We will use fdisk <logical disk device name> to manage and create a new partition on to the disk. For our example, type "fdisk /dev/nvme0n2".
5. In fdisk, type 'm' to view the available options.
6. Type "p" to list the existing partitions on the hard disk.

```
[student@client ~]$ sudo fdisk /dev/nvme0n2
Welcome to fdisk (util-linux 2.32.1).
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 0x334e2152.

Command (m for help): p
Disk /dev/nvme0n2: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x334e2152

Command (m for help): █
```

7. To create a new partition, type 'n'.
Follow by typing a 'p' to create a primary partition.
Type "1" for Partition number 1.
Press enter to accept the default starting sector.
Press enter to accept the default ending sector.
This will use the entire disk to be your partition 1.

```
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 (2048-2097151, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-2097151, default 2097151):

Created a new partition 1 of type 'Linux' and of size 1023 MiB.

Command (m for help): █
```

Press Enter

Press Enter

Type 'p' to list the partition info. Note your first partition and its device name (/dev/nvme0n2p1). Also note its Disk identifier, and the Device ID, which is 83 for Linux.

```

Command (m for help): p
Disk /dev/nvme0n2: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x334e2152

Device            Boot Start      End Sectors  Size Id Type
/dev/nvme0n2p1    2048 2097151 2095104 1023M 83 Linux

Command (m for help):

```

Type 't' to change the partition 1 System type to Linux LVM. (using hex code : 8e)

```

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'.

Command (m for help): p
Disk /dev/nvme0n2: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x334e2152

Device            Boot Start      End Sectors  Size Id Type
/dev/nvme0n2p1    2048 2097151 2095104 1023M 8e Linux LVM

Command (m for help):

```

Type 'w' to write changes to disk and exit fdisk.

```

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

[student@client ~]$

```

8. **Restart** your system for the changes in the partition table to take effect.
9. After the system restart login to the system as root. Type "`fdisk -l <logical disk device name>`" to view and verify the new partition table.

Create LVM components: pvcreate, vgcreate and lvcreate

At this point you have created an empty LVM partition on the new disk device.

On Client (After restarted) :

Although we have created a new LVM partition in the /dev/nvmen02 disk, this LVM is not usable until we have prepared it by using the following steps:

1. Create a physical volume onto the new Linux LVM partition. Type:

```
pvccreate <partition device name>
```

(you can run `lsblk` to find out the partition device name)

```
[student@client ~]$ lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0              11:0    1 1024M  0 rom
nvme0n1         259:0    0   28G  0 disk
├─nvme0n1p1     259:1    0    1G  0 part /boot
├─nvme0n1p2     259:2    0   27G  0 part
│   └─ol-root   252:0    0 24.2G  0 lvm  /
│       └─ol-swap 252:1    0   2.8G  0 lvm  [SWAP]
└─nvme0n2       259:3    0    1G  0 disk
    └─nvme0n2p1 259:4    0 1023M  0 part
```

From the above, the partition device name is **nvme0n2p1** in our case.

```
[student@client ~]$ sudo pvccreate /dev/nvme0n2p1
[sudo] password for student:
Physical volume "/dev/nvme0n2p1" successfully created.
[student@client ~]$
```

The above shown how to create the physical volume.

2. Create a logical volume group onto the physical volume and give it a label (vg name). Type:

```
vgcreate <vg name> <partition device name>
```

For instance, we will use `vg_xfs1` as the vg name for this exercise.

```
[student@client ~]$ sudo vgcreate vg_xfs1 /dev/nvme0n2p1
Volume group "vg_xfs1" successfully created
[student@client ~]$ █
```

3. Display the newly created logical volume group information to find the effective free space available in this vg. Type:

```
vgdisplay <vg name>
```



```
[student@client ~]$ sudo vgdisplay vg_xfs1
--- Volume group ---
VG Name                vg_xfs1
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                1020.00 MiB
PE Size                4.00 MiB
Total PE               255
Alloc PE / Size        0 / 0
Free PE / Size         255 / 1020.00 MiB
VG UUID                S32Gcd-d52t-gQms-ORxd-YRf9-W5Ge-FkFnN0

[student@client ~]$ █
```

For the above, we have 255 physical extents or 1020 MB free space available in `vg_xfs`. This volume group has been assigned with a unique VG UUID too.

4. Finally, we may create a logical volume based on the `vg` and give it a label (lv name). A linux filesystem can be formatted(built) onto this lv later. Type:

```
lvcreate -L +1020M -n laslv vg_xfs1
```

or

```
lvcreate --extents 255 -n laslv vg_xfs1
```

```
[student@client ~]$ sudo lvcreate --extents 255 -n laslv vg_xfs1
Logical volume "laslv" created.
[student@client ~]$
[student@client ~]$ █
```

5. Before moving on to the next section, use `lvdisplay` command to verify the logical device name of the newly created logical volume (lv) Type:

```
lvdisplay /dev/<vg name>/<lv name>
```

For instance, the `vg` name is `vg_xfs1`, and the `lv` name is `laslv`:

```
[student@client ~]$ sudo lvdisplay /dev/vg_xfs1/laslv
--- Logical volume ---
LV Path                /dev/vg_xfs1/laslv
LV Name                laslv
VG Name                vg_xfs1
LV UUID                nHGhZr-F3Po-0EPL-s0HY-DIp2-jJdQ-qDHBZW
LV Write Access        read/write
LV Creation host, time client.example.com, 2022-10-03 15:30:00 +0800
LV Status              available
# open                 0
LV Size                1020.00 MiB
Current LE             255
Segments               1
Allocation              inherit
Read ahead sectors     auto
  - currently set to   8192
Block device           252:2

[student@client ~]$
```

Format partition with XFS filesystem

From this point onward we can use the logical lv device name to carry out the file system creation operation.

1. Format the XFS* filesystem on the LVM partition (e.g. /dev/vg_xfs1/laslv). Type:

```
mkfs -t xfs /dev/vg_xfs1/laslv
```

** XFS is the default and recommended file system on Oracle Linux.*

You may see output like the following :

```
[student@client ~]$ sudo mkfs -t xfs /dev/vg_xfs1/laslv
meta-data=/dev/vg_xfs1/laslv isize=512    agcount=4, agsize=65280 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=0 inobtcount=0
data      =                       bsize=4096   blocks=261120, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming    =version 2              bsize=4096   ascii-ci=0, ftype=1
log       =internal log          bsize=4096   blocks=1566, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime  =none                  extsz=4096   blocks=0, rtextents=0
[student@client ~]$
```

2. As root user, create a mount point and mount the virtual block device. Type

```
mkdir /filesys1
```

3. Try to mount the filesystem onto /filesys1. Type

```
mount /dev/vg_xfs1/laslv /filesys1
```

4. Change directory to /filesys1 and create a testing file there.

```
[student@client ~]$ sudo -i
[root@client ~]# mkdir /filesys1
[root@client ~]# mount /dev/vg_xfs1/laslv /filesys1
[root@client ~]# cd /filesys1
[root@client filesys1]# touch helloworld
[root@client filesys1]# ls -l
total 0
-rw-r--r--. 1 root root 0 Oct  3 15:37 helloworld
[root@client filesys1]#
```

5. Change to / directory. Unmount the /dev/vg_xfs1/laslv file system from its mount point and you will not see the helloworld file in /filesys1 anymore.

```
[root@client filesys1]# cd
[root@client ~]# ls /filesys1
helloworld
[root@client ~]# umount /dev/vg_xfs1/laslv
[root@client ~]# ls /filesys1
[root@client ~]#
```

Take note that, the helloworld file is still existing. We cannot see it only because the file system has been unmounted. If we mount the file system again, we will see the file.

Mounting an XFS filesystem to the system permanently

We need to know the uuid of the file system in the logical volume to proceed.

1. Use the blkid command to find the corresponding block device UUID (Universal Unique Identifier) of the target **filesystem**. Type:

```
blkid /dev/vg_xfs1/laslv
```

For instance:

```
[root@client ~]# blkid /dev/vg_xfs1/laslv
/dev/vg_xfs1/laslv: UUID="f2ca68eb-3b22-4cad-a910-6a8cef555e4e" BLOCK_SIZE="512" TYPE="xfs"
[root@client ~]#
```

2. Append an entry to /etc/fstab (in a single line) so that the new filesystem is automatically mounted on bootup. Replace the UUID value with the UUID of your new filesystem. The double quotes around the UUID are optional.

```
UUID="f2ca68eb-3b22-4cad-a910-6a8cef555e4e" /filesys1 xfs defaults 0 0
```

```

GNU nano 2.9.8 /etc/fstab
#
# /etc/fstab
# Created by anaconda on Sun Sep 11 10:55:22 2022
#
# 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.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/ol-root / xfs defaults 0 0
UUID=c008dd7e-f7ec-40fc-bf13-ff107ebe6a8c /boot xfs defaults 0 0
/dev/mapper/ol-swap none swap defaults 0 0
UUID=f2ca68eb-3b22-4cad-a910-6a8cef555e4e /filesys1 xfs defaults 0 0

[ Read 15 lines ]
^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line

```

(Please be **very careful** when you are editing the /etc/fstab file. It is an important system configuration file.)

- The new filesystem will be mounted automatically every time your system is started up.
In fact, right after you have update the /etc/fstab file, you have the option to mount and unmount the filesystem using the mount point directory name instead of the block device name (ie. /dev/vg_xfs/lv_anyname).

Try:

```

mount /filesys1
and/or
umount /filesys1

```

- Test that you can locate the helloworld file in /filesys1.

```

[root@client ~]# mount /filesys1
[root@client ~]# ls -l /filesys1
total 0
-rw-r--r--. 1 root root 0 Oct 3 15:37 helloworld
[root@client ~]# umount /filesys1
[root@client ~]# ls -l /filesys1
total 0
[root@client ~]#

```

- Restart your Client VM. Use the df command to display the current storage usage. Type:

```
df
```

```

[student@client ~]$ df
Filesystem                1K-blocks    Used Available Use% Mounted on
devtmpfs                   912760         0    912760   0% /dev
tmpfs                      942460         0    942460   0% /dev/shm
tmpfs                      942460    9684    932776   2% /run
tmpfs                      942460         0    942460   0% /sys/fs/cgroup
/dev/mapper/ol-root        25358236 6909720  18448516  28% /
/dev/mapper/vg_xfs1-laslv  1038216   40280   997936    4% /filesys1
/dev/nvme0n1p1             1038336  302604   735732  30% /boot
tmpfs                      188492         12   188480   1% /run/user/42
tmpfs                      188492         20   188472   1% /run/user/1000
[student@client ~]$

```

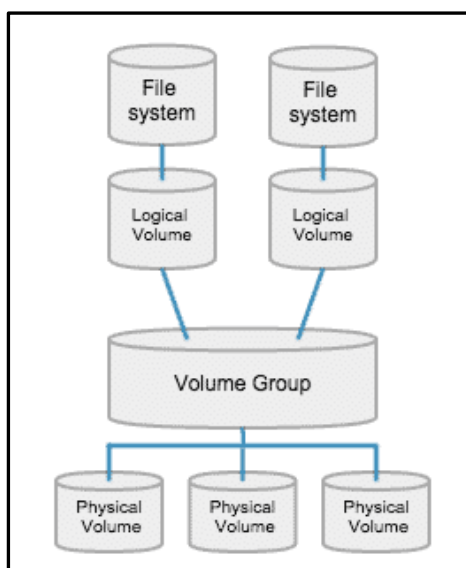
df command reports all the file systems and their disk usage information.
From the above, observe the three disk related file systems: /, /filesys1 and /boot.
The report also shows the logical device names for each of the corresponding file systems.

Note: Filesystem that built on top of a LVM is much flexible for future expansions. It is generally recommended for server based systems. For desktop based Linux system, LVM may not be necessary as it takes up more resource to be running.
For Oracle Linux 8.6. Both of the LVM and the XFS are set to be used by default.

6. You may comment out the /filesys1 entry from the /etc/fstab file after the completion of all the above exercises.

Basic Structure and elements of LVM

At the end, refer to the following diagram and check you can identify and associate the illustrated elements with the operations you have done in the exercises.



Source : <https://cdn.thegeekdiary.com/wp-content/uploads/2014/10/LVM-basic-structure.png>

Challenge Exercise (Optional)

Try to add one more virtual disk (1Gb) to your client VM and extend your /dev/mapper/vg_xfs1-laslv file system with the additional disk storage.

```

[root@client ~]# lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0                                  11:0    1 1024M  0 rom
nvme0n1                             259:0    0   28G  0 disk
├─nvme0n1p1                         259:1    0    1G  0 part /boot
├─nvme0n1p2                         259:2    0   27G  0 part
│   └─ol-root                       252:0    0  24.2G  0 lvm /
│       └─ol-swap                   252:1    0    2.8G  0 lvm [SWAP]
└─nvme0n2                             259:3    0    1G  0 disk
    ├─nvme0n2p1                     259:4    0 1023M  0 part
    │   └─vg_xfs1-laslv             252:2    0    2G  0 lvm /filesys1
    └─nvme0n3                             259:5    0    1G  0 disk
        ├─nvme0n3p1                 259:6    0 1023M  0 part
        └─vg_xfs1-laslv             252:2    0    2G  0 lvm /filesys1

[root@client ~]# df -h /filesys1
Filesystem                Size      Used Avail Use% Mounted on
/dev/mapper/vg_xfs1-laslv 2.0G       47M    2.0G   3% /filesys1
[root@client ~]#
  
```

As shown in the above, another 1Gb disk 'nvme0n3' has been added to the system. With the additional disk storage, the /dev/mapper/vg_xfs1-laslv file system has extended its size to 2.0Gb.

2. Filesystem Access Control Lists

Filesystem Access Control Lists (ACLs) are available for a variety of Linux filesystems including ext2, ext3, ext4, and XFS. Filesystem ACLs are extremely handy in that they allow you to extend access controls to files and directories beyond the simple user/group/other ownership.

For older filesystems such as ext2 - ext4, they require some additional configuration (or even kernel patches) to enable the ACLs. For XFS filesystem, the ACL feature is enabled by default.

Note: The native Linux file/directory access control scheme only allows read, write and execute permissions to be set at **owner**, **group** and **others**. This simple scheme has many limitations to implement certain common required permission assignment. For example, it is not easy to allow multiple groups to access to a file. The workaround may lead to a creation of a new group and place selected users to join to this group and with it granted for the file access permission. With filesystem ACLs, we can create additional access control list on top of the native permission scheme to allow users belong to different groups to gain access to the same file.

The getfacl and setfacl commands

There are only two new commands involved with the ACLs permissions.

On Server:

1. Login as root and create a folder /var/test_acl. Type:

```
mkdir /var/test_acl
```

2. Examine the default ACLs for the newly created folder /var/test_acl. Type:

```
cd /var  
getfacl test_acl
```

```
[root@server ~]# mkdir /var/test_acl  
[root@server ~]# cd /var  
[root@server var]# getfacl test_acl  
# file: test_acl  
# owner: root  
# group: root  
user::rwx  
group::r-x  
other::r-x  
  
[root@server var]# █
```

From the above, we learn that the group id of the folder is 'root' and this folder allows any users to read from and access to it.

You may wonder why we need to change directory (cd) to /var before running the getfacl command. Why not simply type:

```
getfacl /var/testacl
```

For your reference:

```
[root@server ~]# getfacl /var/test_acl
getfacl: Removing leading '/' from absolute path names
# file: var/test_acl
# owner: root
# group: root
user::rw-
group::r-x
other::r-x

[root@server ~]#
```

As shown at the above, getfacl command, by default, removes the leading '/' from the command line argument as it prefers to work with relative path rather than absolute path.

This default setting is to avoid the case of "Absolute path used when a relative path should have been used. This is what this restriction is designed to highlight to have strong sense of security in mind.

3. Disable all permissions from the 'other' users for the /var/test_acl directory and check the effective permissions with getfacl. Type:

```
chmod o= /var/test_acl
getfacl -p /var/test_acl
```

```
[root@server ~]# ls -ld /var/test_acl
drwxr-xr-x. 2 root root 6 Oct 3 17:49 /var/test_acl
[root@server ~]# chmod o= /var/test_acl
[root@server ~]# getfacl -p /var/test_acl
# file: /var/test_acl
# owner: root
# group: root
user::rw-
group::r-x
other::---

[root@server ~]#
```

Note: The `ls` command with `-ld` options list the details of a directory. From the above, the initial permissions given to 'other' were read and execute (for directory, execute means accessible.) After the `chmod` command the effective permissions have changed accordingly. Take note that the `-p` option is to specify the `getfacl` should take absolute path as its argument. Thus, you do not see the warning message : "Removing leading '/'".

4. Verify that other users (not in the 'root' group) are not able to access to `/var/test_acl`.

```
[root@server ~]# su - peter
[peter@server ~]$ cd /var/test_acl
-bash: cd: /var/test_acl: Permission denied
[peter@server ~]$ exit
logout
[root@server ~]# su - student
[student@server ~]$ ls -l /var/test_acl
ls: cannot open directory '/var/test_acl': Permission denied
[student@server ~]$ ls -ld /var/test_acl
drwxr-x---. 2 root root 6 Oct  3 17:49 /var/test_acl
[student@server ~]$ exit
logout
[root@server ~]#
```

For instance, the above shows neither peter nor student can access to the folder `/var/test_acl` (using `cd` nor `ls -l`). However, `ls -ld` is okay. **why?**

5. Enable the group members of `ppm` to have read, write and access permission to the `/var/test_acl` using ACLs. Type :

```
setfacl -m g:ppm:rwX /var/test_acl
```

```
[root@server ~]# setfacl -m g:ppm:rwX /var/test_acl
[root@server ~]# getfacl -p /var/test_acl
# file: /var/test_acl
# owner: root
# group: root
user::rwX
group::r-x
group:ppm:rwX
mask::rwX
other::---

[root@server ~]# ls -ld /var/test_acl/
drwxrwX---+ 2 root root 6 Oct  3 17:49 /var/test_acl/
[root@server ~]#
```

From the above, you can see the group members of 'ppm' is now having the `rwX` permissions to the `/var/test_acl` directory. Moreover, the `ls -ld` cannot show the detail / exact permissions of the directory. However, the '+' symbol helps to indicate there is extended file ACL associating with the item.

6. Verify that users in the `ppm` group (peter, paul and mary) are able to access to

`/var/test_acl` directory and create new files in it. On the other hand, the directory remains not accessible by the student user.

```
[root@server ~]# su - student
[student@server ~]$ touch /var/test_acl/student
touch: cannot touch '/var/test_acl/student': Permission denied
[student@server ~]$ exit
logout
[root@server ~]# su - peter
[peter@server ~]$ touch /var/test_acl/peter
[peter@server ~]$ exit
logout
[root@server ~]# su - paul
[paul@server ~]$ touch /var/test_acl/paul
[paul@server ~]$ exit
logout
[root@server ~]# su - mary
[mary@server ~]$ touch /var/test_acl/mary
[mary@server ~]$ exit
logout
[root@server ~]# ls -l /var/test_acl/
total 0
-rw-rw-r--. 1 mary  mary  0 Oct  4 10:42 mary
-rw-rw-r--. 1 paul  paul  0 Oct  4 10:42 paul
-rw-rw-r--. 1 peter peter 0 Oct  4 10:42 peter
[root@server ~]#
```

7. `setfacl` can also be used to set permission to '**deny**' specific users or groups to access to a file/directory by using 'MASK' patterns.

To deny mary to have the any permissions on `/var/test_acl` using `setfacl`. Type:

```
setfacl -m u:mary:000 /var/test_acl
```

(The 000 is the mask value represents no read, no write nor no execute.)

```
[root@server ~]# setfacl -m u:mary:000 /var/test_acl
[root@server ~]# getfacl -p /var/test_acl
# file: /var/test_acl
# owner: root
# group: root
user::rwx
user:mary:---
group::r-x
group:ppm:rwx
mask::rwx
other::---
```

8. Verify that users in the ppm group (peter and paul except mary) are able to access to `/var/test_acl` directory and create new files in it.

```
[root@server ~]# su - mary
[mary@server ~]$ cd /var/test_acl/
-bash: cd: /var/test_acl/: Permission denied
[mary@server ~]$ exit
logout
[root@server ~]# su - peter
[peter@server ~]$ cd /var/test_acl
[peter@server test_acl]$ exit
logout
[root@server ~]# su - paul
[paul@server ~]$ cd /var/test_acl/
[paul@server test_acl]$ exit
logout
[root@server ~]#
```

9. That's all for the ACLs. To end this section of exercises The last command we will try is to remove all the ACLs entries from the /var/test_acl. Type:

```
setfacl -b /var/test_acl
```

-b option remove all the ACLs from the target file/directory.

```
[root@server ~]# setfacl -b /var/test_acl
[root@server ~]# getfacl -p /var/test_acl/
# file: /var/test_acl/
# owner: root
# group: root
user::rwx
group::r-x
other::---

[root@server ~]# ls -ld /var/test_acl/
drwxr-x---. 2 root root 43 Oct  4 10:42 /var/test_acl/
[root@server ~]#
```

3. Exporting directories on the NFS Server - for remote files access

"Network File System (NFS) is a distributed file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a computer network much like local storage is accessed."

~https://en.wikipedia.org/wiki/Network_File_System

In this and the following sections, we will explore the usages of NFS by configuring our Server to run NFS services.

On server:

1. Check if the nfs package has been installed:

```
dnf info nfs-utils
(Install it if needed)
```

2. Start the nfs service if it is not running yet.

```
systemctl status nfs-server
systemctl start nfs-server
```

```
[root@server ~]# systemctl status nfs-server.service
● nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; vendor pr>
   Active: inactive (dead)
[root@server ~]#
[root@server ~]# systemctl start nfs-server
[root@server ~]# systemctl status nfs-server
● nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; vendor pr>
   Active: active (exited) since Tue 2022-10-04 11:02:22 +08; 6s ago
     Process: 5028 ExecStart=/bin/sh -c if systemctl -q is-active gssproxy; then syst>
     Process: 5017 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
     Process: 5015 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
    Main PID: 5028 (code=exited, status=0/SUCCESS)

Oct 04 11:02:22 server.example.com systemd[1]: Starting NFS server and services...
Oct 04 11:02:22 server.example.com systemd[1]: Started NFS server and services.
[root@server ~]#
[root@server ~]#
```

3. Create a directory /exports/data and make it world-writable.

```
mkdir -p /exports/data
chmod 777 /exports/data
```

4. Create some files in /exports/data.

```
cd /exports/data
touch a b c
```

```
[root@server ~]# mkdir -p /exports/data
[root@server ~]# chmod 777 /exports/data
[root@server ~]# cd /exports/data
[root@server data]# touch a b c
[root@server data]# ls -l
total 0
-rw-r--r--. 1 root root 0 Oct  4 11:13 a
-rw-r--r--. 1 root root 0 Oct  4 11:13 b
-rw-r--r--. 1 root root 0 Oct  4 11:13 c
[root@server data]#
```

Use -p switch in the mkdir command is to ensure the targeted folder will be created.

5. Edit **/etc/exports** to have the following line, replacing *clientIP* with the IP address of your client.

```
/exports/data clientIP(ro,sync)
```



Warning

The format of the `/etc/exports` file is very precise, particularly in regards to use of the space character. Remember to always separate exported file systems from hosts and hosts from one another with a space character. However, there should be no other space characters in the file except on comment lines.

```
GNU nano 2.9.8 /etc/exports Mc
/exports/data 192.168.30.130(ro,sync)
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To L
```

(The sample ClientIP is 192.168.30.130)

6. Run `exportfs` with `-r` option to re-export the entries in `/etc/exports`.

```
exportfs -r
```

7. Run `exportfs` with `-v` option to check the exports.

```
exportfs -v
```

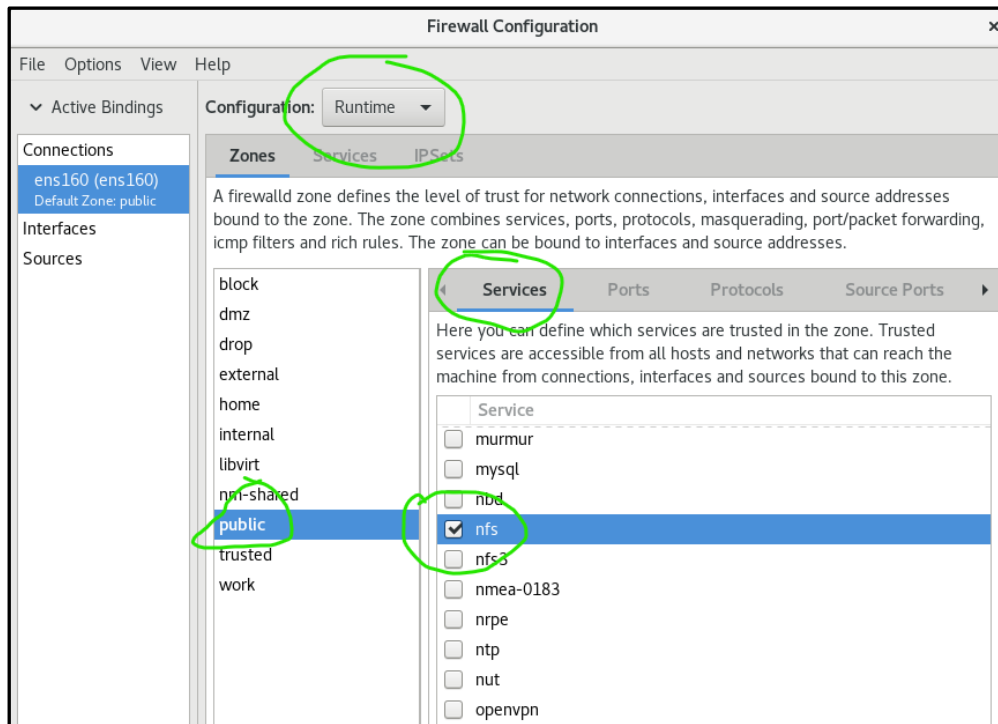
```
[root@server ~]# exportfs -r
[root@server ~]# exportfs -v
/exports/data 192.168.30.130(sync,wdelay,hide,no_subtree_check,sec=sys,fsync,secure,
root_squash,no_all_squash)
[root@server ~]#
```

Take note that, there are many default options have been defined for this exported NFS folder. The `ro` (ReadOnly) and `sync` options we have defined in the `/etc/exports` configuration file are in effect too.

8. Adjust the firewall* to allow connections to NFS on TCP Port 2049. Also allow connections to the loopback interface (if you are using the Firewall GUI to configure the firewall, access to the loopback interface will be automatically enabled)

[* depending on the version of the `nfs-utils` you may need to enable more services - ie. `nfs`, `nfs3`, in other system. For our setup, we are running `nfs4`, TCP Port 2049 is

sufficient.]



4. Mounting exported directories on the NFS Client

In the previous section, we have enabled a NFS service at the server to allow the client to access to the /exports/data folder over the network with NFS protocol.

Based on the operation model, to access to remote NFS based content, we need to mount the remote NFS (exported folder) as a file system to the local system, the client.

On client:

1. Check the installed nfs package version

```
dnf list nfs-utils
```

```
[root@client ~]#
[root@client ~]# dnf list nfs-utils
Last metadata expiration check: 1:22:22 ago on Tue 04 Oct 2022 10:03:43 AM +0
Installed Packages
nfs-utils.x86_64      1:2.3.3-51.el8      @anaconda
[root@client ~]#
```

(The NFS version looks good as it is the same as the one installed at the server.)

2. Create a new empty directory at the client as a mount point for the remote NFS. Type

```
mkdir -p /mount/data
```

3. Mount* the exported directory.

```
mount <serverIP>:/exports/data /mount/data -o rw
```

*The above may not work if the versions of the installed nfs packages are not the compatible with the one at the server.

(Note: The serverIP in our example is 192.168.30.88. Your serverIP may not be the same.)

```
[root@client ~]# mkdir -p /mount/data
[root@client ~]# mount 192.168.30.88:/exports/data /mount/data -o rw
[root@client ~]# ls -l /mount/data
total 0
-rw-r--r--. 1 root root 0 Oct  4 11:13 a
-rw-r--r--. 1 root root 0 Oct  4 11:13 b
-rw-r--r--. 1 root root 0 Oct  4 11:13 c
[root@client ~]#
```

The above shows the mount operation is successful, and we can see the remote files a, b, and c under the /mount/data directory. This local directory is now mounted with a remote filesystem. We can access to the remote file content seamlessly as if it is our local file content.

(Note: in case your mount is failed, check the following 2 configurations: Your Server Firewall has openend nfs services (or TCP port 2049), and your server IP is correctly specified.)

4. Check the contents of /mount/data. Try to create a file in /mount/data. You should not be successful. You should only have read-only access to the mounted directory because it was exported with the ro option.

```
[root@client ~]# cd /mount/data
[root@client data]# touch d e f
touch: cannot touch 'd': Read-only file system
touch: cannot touch 'e': Read-only file system
touch: cannot touch 'f': Read-only file system
[root@client data]#
```

5. Run the mount command without any options nor arguments to see all the mounted devices. By redirecting the output of the mount to a grep command with the pipe (|), we can display the only mount entry that related to nfs4 (NFS version 4).

```
mount | grep nfs4
```

```
[root@client ~]# mount | grep nfs4
192.168.30.88:/exports/data on /mount/data type nfs4 (rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=192.168.30.130,local_lock=none,addr=192.168.30.88)
[root@client ~]#
```

The above show a few interesting and important attributes of the NFS file system. The attribute shows that the file system is rw (read and write) but it is in fact read only.

It is because, the 'rw' showing at the above is the default mount option (always assume rw). The effective access is controlled by the remote NFS itself (based on its export option settings.) To have a read and write NFS service, both the remote export option and the local mount option must be set to read and write (rw).

6. Unmount the nfs file system for now. Type:

```
umount /mount/data
```



```
[root@client ~]# cd
[root@client ~]# pwd
/root
[root@client ~]# ls /mount/data
a b c
[root@client ~]# umount /mount/data
[root@client ~]# ls /mount/data
[root@client ~]# mount | grep nfs4
[root@client ~]#
```

The above screenshot demonstrates the files: a, b and c will not be accessible as the NFS file system has been unmounted. Take note that, we have to cd to some where outside the mount point directory, to unmount the file system. If not, you will receive a device is busy error message:

```
[root@client ~]# mount 192.168.30.88:/exports/data /mount/data -o rw
[root@client ~]# cd /mount/data
[root@client data]# ls
a b c
[root@client data]# umount /mount/data
umount.nfs4: /mount/data: device is busy
[root@client data]#
```

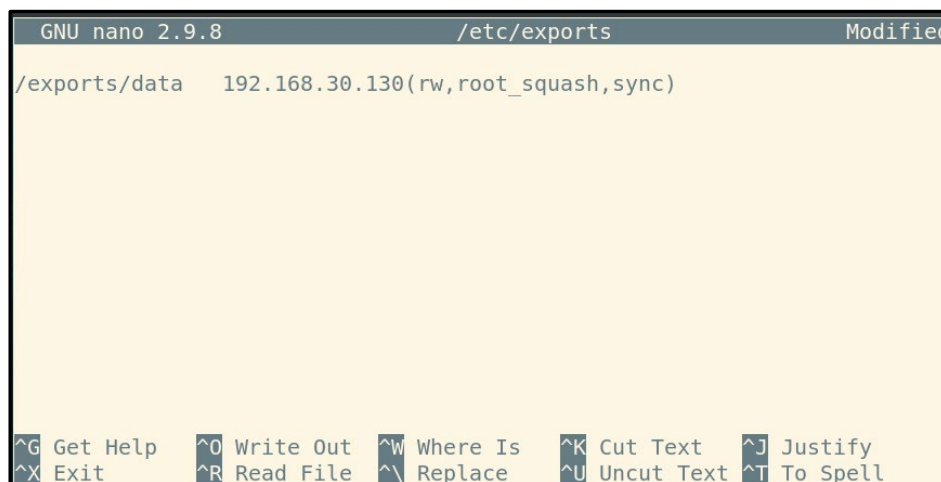
5. NFS Export options

In the previous section, we have used ro and sync options to define the NFS attributes. We will explore a couple more NFS export options in this section.

On server:

1. Edit /etc/exports and change the export options to the following:

```
/exports/data    clientIP(rw,root_squash,sync)
```



```
GNU nano 2.9.8 /etc/exports Modified
/exports/data 192.168.30.130(rw,root_squash,sync)

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell
```

2. Run exportfs with -r option to re-export the entries in /etc/exports.
3. Run exportfs with -v option to check the exports.

```
[root@server ~]# exportfs -r
[root@server ~]# exportfs -v
/exportfs/data 192.168.30.130(sync,wdelay,hide,no_subtree_check,sec=sys,
rw,secure,root_squash,no_all_squash)
[root@server ~]#
```

Note:

root_squash option - is enabled by default. With this option, NFS shares change the remote root user to the nobody user, an unprivileged user account. This changes the owner of all remote root-created files to **nobody**, which prevents a remote root user uploading of programs with the setuid bit set.

If no_root_squash is used, remote root users can change any file on the shared file system and leave applications infected by Trojans for other users to inadvertently execute.

sync option - If the sync option is specified on a mount point, any system call that writes data to files on that mount point causes that data to be flushed to the server before the system call returns control to user space. This provides greater data cache coherence among clients, but at a significant performance cost. Default setting is sync option disabled. Sync option is recommended when there are multiple clients mounting the same NFS concurrently.

On client :

4. Make sure no one is accessing the mounted directory /mount/data.
5. Make sure there is no filesystem mounted using /mount/data as a mount point. Type:

```
umount /mount/data
```

```
[root@client ~]# cd /
[root@client /]# umount /mount/data
umount: /mount/data: not mounted.
[root@client /]#
```

6. Mount the exported directory again.

```
mount serverIP:/exports/data /mount/data
```

```
[root@client ~]# mount 192.168.30.88:/exports/data /mount/data
[root@client ~]# ls -l /mount/data/
total 0
-rw-r--r--. 1 root root 0 Oct  4 11:13 a
-rw-r--r--. 1 root root 0 Oct  4 11:13 b
-rw-r--r--. 1 root root 0 Oct  4 11:13 c
[root@client ~]# mount | grep nfs4
192.168.30.88:/exports/data on /mount/data type nfs4 (rw,relatime,vers=4.2,rsz=262144,wsz=262144,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=192.168.30.130,local_lock=none,addr=192.168.30.88)
[root@client ~]#
```


The mount is successful, and we can see the a, b and c files. The default rw option is also set even we did not specify it in the mount command.

- As root, create a new file in /mount/data. Name it 'byclientRoot.txt'. You should be successful as you now have read-write access. Observe the owner and group id of the newly created file.

```
[root@client ~]# touch /mount/data/byclientRoot.txt
[root@client ~]# ls -l /mount/data/
total 0
-rw-r--r--. 1 root    root    0 Oct  4 11:13 a
-rw-r--r--. 1 root    root    0 Oct  4 11:13 b
-rw-r--r--. 1 nobody  nobody  0 Oct  4 17:23 byclientRoot.txt
-rw-r--r--. 1 root    root    0 Oct  4 11:13 c
[root@client ~]#
```

Take note from the above, the user and group ids of the newly created file are both set to 'nobody'. This is the effect of 'root_squash' export option.

The client root user over the NFS The user root is mapped to the user nobody when accessing the exported directory.

On client (login as student):

- As user student, create another file in /mount/data, name it byclientStudent.txt . Type:

```
touch /mount/data/byclientStudent.txt
```

- View the file user and group id and the file permissions of these newly created files.

```
ls -l /mount/data
```

```
[root@client ~]# su - student
[student@client ~]$ touch /mount/data/byclientStudent.txt
[student@client ~]$ ls -l /mount/data
total 0
-rw-r--r--. 1 root    root    0 Oct  4 11:13 a
-rw-r--r--. 1 root    root    0 Oct  4 11:13 b
-rw-r--r--. 1 nobody  nobody  0 Oct  4 17:23 byclientRoot.txt
-rw-rw-r--. 1 student student 0 Oct  4 17:25 byclientStudent.txt
-rw-r--r--. 1 root    root    0 Oct  4 11:13 c
[student@client ~]$
```

This time, there is no 'squash' effect applied, as student is not the root.

For the user nobody, its default file permission mask is different (more conservative) from normal user account.

Thus, the two files created by root (squashed to nobody) previously and by student have different default file permissions.

6. Linux File Permissions vs NFS Permissions

Continue from the last exercise:

On server:

1. Create a subdirectory `/exports/data/student` and change the file and group owner to `student`. Verify that only `student` can write in this directory.

```
mkdir -p /exports/data/student
chown student.student /exports/data/student
ls -ld /exports/data/student
```

2. As user `student`, create some files in `/exports/data/student`.

```
[root@server ~]# mkdir -p /exports/data/student
[root@server ~]# ls -ld /exports/data/student
drwxr-xr-x. 2 root root 6 Oct 5 10:22 /exports/data/student
[root@server ~]# chown student.student /exports/data/student
[root@server ~]# ls -ld /exports/data/student
drwxr-xr-x. 2 student student 6 Oct 5 10:22 /exports/data/student
[root@server ~]# su - student
[student@server ~]$ cd /exports/data/student/
[student@server student]$ ls
[student@server student]$ touch d e f
[student@server student]$ ls -l
total 0
-rw-rw-r--. 1 student student 0 Oct 5 10:24 d
-rw-rw-r--. 1 student student 0 Oct 5 10:24 e
-rw-rw-r--. 1 student student 0 Oct 5 10:24 f
[student@server student]$
```

On client: (Ensure the `/exports/data` has been mounted.)

3. As user `student`, modify the file that you have created in the previous step in `/mount/data/student`. You should be successful. Type:

```
cd /mount/data/student
echo hello world >> f
cat f
```

```
[root@client ~]# su - student
[student@client ~]$ cd /mount/data/student/
[student@client student]$ echo hello world >> f
[student@client student]$ ls -l
total 4
-rw-rw-r--. 1 student student 0 Oct 5 10:24 d
-rw-rw-r--. 1 student student 0 Oct 5 10:24 e
-rw-rw-r--. 1 student student 12 Oct 5 10:35 f
[student@client student]$ cat f
hello world
[student@client student]$
```

4. As user student, create a file in /mount/data/student. You should be successful.
5. As user root, try to access to /mount/data/student. You **should not** be successful.

```
[student@client ~]$ touch /mount/data/student/byRemoteClient
[student@client ~]$ ls -l /mount/data/student
total 4
-rw-rw-r--. 1 student student 0 Oct 5 10:42 byRemoteClient
-rw-rw-r--. 1 student student 0 Oct 5 10:24 d
-rw-rw-r--. 1 student student 0 Oct 5 10:24 e
-rw-rw-r--. 1 student student 12 Oct 5 10:35 f
[student@client ~]$ exit
logout
[root@client ~]# touch /mount/data/student/byRemoteRoot
touch: cannot touch '/mount/data/student/byRemoteRoot': Permission denied
[root@client ~]#
```

The user root is mapped to user nobody and user nobody has no access right to create files in the /mount/data/student folder which belongs to the mounted file system.

6. Special notes on the user mapping between the server and the client systems.

There is a serious issue related to the NFS user mapping operations. So far in our NFS exercises, we may overlook the fact that the user accounts of the server and the client systems are independent to each other's. In the previous exercises, we used the server student to create a file and let the client student to access and update. This operation itself is wrongfully assumed these two accounts are used by the same user. In many cases, this may not be true.

Note: NFS only uses the user and group id to determine the access permissions, if the two systems (server and client) are having the same user id but assigned to different users, the NFS permissions scheme may allow unauthorized file operations. A proper NFS setup in an Enterprise network requires a Name to IDs (user id and Group id) mapping solutions. For instance, idmapd or Kerberos. The id mapper solution is out of the scope for LAS.

For now, you may take note on the following recommendation.

Only run NFS in read only mode to reduce security threat.

Only allow NFS read write operations between systems that have very simple and small user account configurations.

7. To illustrate the impact of id mapping issues:

On server:

Login in as paul in the server, create a file in the /exports/data directory. Type:

```
cd /exports/data
touch byPaul
```

```
[root@server ~]# su - paul
[paul@server ~]$ touch /exports/data/byPaul
[paul@server ~]$ ls -l /exports/data
total 0
-rw-r--r--. 1 root    root    0 Oct  4 11:13 a
-rw-r--r--. 1 root    root    0 Oct  4 11:13 b
-rw-r--r--. 1 nobody  nobody  0 Oct  4 17:23 byclientRoot.txt
-rw-rw-r--. 1 student student 0 Oct  4 17:25 byclientStudent.txt
-rw-rw-r--. 1 paul    paul    0 Oct  5 10:47 byPaul
-rw-r--r--. 1 root    root    0 Oct  4 11:13 c
drwxr-xr-x. 2 student student 55 Oct  5 10:42 student
[paul@server ~]$
```

On Client:

Login as any user, list the directory content of /mount/data. Type:

```
ls -l /mount/data
```

```
[student@client student]$ cd
[student@client ~]$ ls -l /mount/data
total 0
-rw-r--r--. 1 root    root    0 Oct  5 13:54 a
-rw-r--r--. 1 root    root    0 Oct  5 13:54 b
-rw-r--r--. 1 nobody  nobody  0 Oct  5 15:48 byclientRoot
-rw-rw-r--. 1 student student 0 Oct  5 16:02 byclientStudent
-rw-rw-r--. 1 1002    1002    0 Oct  5 16:28 byPaul
-rw-r--r--. 1 root    root    0 Oct  5 13:54 c
drwxr-xr-x. 2 student student 55 Oct  5 16:31 student
[student@client ~]$
```

As show in the above. The user id of paul (1002) does not map to any existing users in the client system, thus the ls -l display the raw user id. **It may not be the worst case; the worst case is when 1002 maps to a user account at the client system and this user account should not be authorized to view or update paul's file.**

8. Unmount the directory (requires root)

```
umount /mount/data
```

7. Accessing exported directories by mounting pseudo-root

Mounting with pseudo-root can help to simplify the NFS mount operations.

On server:

1. Create another directory /exports2/mydata.

```
mkdir -p /exports2/mydata
```

2. Create some files in /exports2/mydata. (see the screen shots below.)

```
[root@server ~]# mkdir -p /exports2/mydata
[root@server ~]# cd /exports2/mydata
[root@server mydata]# touch x y z
[root@server mydata]# ls -l /exports2/mydata
total 0
-rw-r--r--. 1 root root 0 Oct  5 18:17 x
-rw-r--r--. 1 root root 0 Oct  5 18:17 y
-rw-r--r--. 1 root root 0 Oct  5 18:17 z
[root@server mydata]#
```

3. Edit /etc/exports and add the following line.

```
/exports2/mydata *(ro,sync)
```

```
GNU nano 2.9.8 /etc/exports Modified
/exports/data 192.168.30.130(rw,root_squash,sync)
/exports2/mydata *(ro,sync)

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

The /etc/exports file should contain two entries now. The newly added entry is to offer the /exports2/mydata to any client (The * is a wild card) to mount it with read only, root_squash(default), and sync options.

4. Run exportfs with -r and -v options to re-export the entries in /etc/exports. Type:

```
exportfs -rv
```

```
[root@server ~]# exportfs -rv
exporting 192.168.30.130:/exports/data
exporting */exports2/mydata
[root@server ~]#
```

Note: Ensure your nfs server is running and the firewall setting is properly set to allow nfs. (If your have restarted your server, the runtime configuration of the firewall will be reset.)

On client:

Ensure you have unmounted the previously mounted NFS file system.

5. Run the following command to mount the pseudo-root of the server using /mount as the

mount point. Type:

```
mount serverIP:/ /mount
```

Note: In the previous nfs mount commands used, we specified the actual folder path to be mounted: ie.

```
mount serverIP:/exports/data /mount/data -o rw
```

In this exercise, we only use serverIP:/ as the target mount folder (pseudo root).

Use `ls -l` and `mount` to find out what has been mounted from the server.

```
ls -l /mount
mount | grep nfs4
```

```
[root@client ~]# mount 192.168.30.88:/ /mount
[root@client ~]# ls -l /mount
total 0
drwxr-xr-x. 3 root root 18 Oct 4 11:13 exports
drwxr-xr-x. 3 root root 20 Oct 5 10:56 exports2
[root@client ~]# mount | grep nfs4
192.168.30.88:/ on /mount type nfs4 (rw,relatime,vers=4.2,rsz=262144,wsz=262144,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=192.168.30.130,local_lock=none,addr=192.168.30.88)
192.168.30.88:/exports on /mount/exports type nfs4 (rw,relatime,vers=4.2,rsz=262144,wsz=262144,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=192.168.30.130,local_lock=none,addr=192.168.30.88)
192.168.30.88:/exports2 on /mount/exports2 type nfs4 (rw,relatime,vers=4.2,rsz=262144,wsz=262144,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=192.168.30.130,local_lock=none,addr=192.168.30.88)
[root@client ~]#
```

6. Verify the content and access permissions of these two remote directories.

```
[root@client ~]# ls -l /mount/exports/data
total 0
-rw-r--r--. 1 root root 0 Oct 5 13:54 a
-rw-r--r--. 1 root root 0 Oct 5 13:54 b
-rw-r--r--. 1 nobody nobody 0 Oct 5 15:48 byclientRoot
-rw-rw-r--. 1 student student 0 Oct 5 16:02 byclientStudent
-rw-rw-r--. 1 1002 1002 0 Oct 5 16:28 byPaul
-rw-r--r--. 1 root root 0 Oct 5 13:54 c
drwxr-xr-x. 2 student student 55 Oct 5 16:31 student
[root@client ~]# touch /mount/exports/data/remoteFile
[root@client ~]# ls -l /mount/exports/data
total 0
-rw-r--r--. 1 root root 0 Oct 5 13:54 a
-rw-r--r--. 1 root root 0 Oct 5 13:54 b
-rw-r--r--. 1 nobody nobody 0 Oct 5 15:48 byclientRoot
-rw-rw-r--. 1 student student 0 Oct 5 16:02 byclientStudent
-rw-rw-r--. 1 1002 1002 0 Oct 5 16:28 byPaul
-rw-r--r--. 1 root root 0 Oct 5 13:54 c
-rw-r--r--. 1 nobody nobody 0 Oct 5 18:30 remoteFile
drwxr-xr-x. 2 student student 55 Oct 5 16:31 student
[root@client ~]#
```

/mount/exports/data is set to read and write permissions.


```
[root@client ~]# ls -l /mount/exports2/mydata
total 0
-rw-r--r--. 1 root root 0 Oct  5 18:17 x
-rw-r--r--. 1 root root 0 Oct  5 18:17 y
-rw-r--r--. 1 root root 0 Oct  5 18:17 z
[root@client ~]# touch /mount/exports2/mydata/remoteFile2
touch: cannot touch '/mount/exports2/mydata/remoteFile2': Read-only file system
[root@client ~]#
```

/mount/exports2/mydata is set to read only.

Conclusion: pseudo-root mounting can help to mount multiple exported directories in one command line.

7. Unmount the directory

```
umount /mount
```

[Note: Depending if you are using nfs 3 or nfs 4. The pseudo-root mounting behaviour may be different]

8. Mount NFS exported directory on bootup with /etc/fstab

On client:

1. Append the following line in /etc/fstab so that the /mount/data will be mounted automatically upon every bootup.

```
serverIP:/exports/data /mount/data nfs defaults 0 0
```

```
GNU nano 2.9.8 /etc/fstab
#
# /etc/fstab
# Created by anaconda on Sun Aug 29 07:52:25 2021
#
# 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.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/ol-root / xfs defaults 0 0
UUID=851a5543-aed4-4572-86ab-2b72a7b16a0a /boot xfs defa
/dev/mapper/ol-swap none swap defaults 0 0
#UUID="1b3c93a6-7b59-4a37-b8a2-78c968c2ce95" /filesys1 xfs defau
192.168.30.88:/exports/data /mount/data nfs defaults 0 0

[ Read 16 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
```

2. Run the following command or restart the system to mount the /mount/data.


```
mount /mount/data
```
3. Check the contents of /mount/data to see if the server's exported directory has been mounted.

For instance:

```
[root@client ~]# mount /mount/data
[root@client ~]# ls -l /mount/data
total 0
-rw-r--r--. 1 root    root    0 Oct  5 13:54 a
-rw-r--r--. 1 root    root    0 Oct  5 13:54 b
-rw-r--r--. 1 nobody  nobody  0 Oct  5 15:48 byclientRoot
-rw-rw-r--. 1 student student 0 Oct  5 16:02 byclientStudent
-rw-rw-r--. 1 1002    1002    0 Oct  5 16:28 byPaul
-rw-r--r--. 1 root    root    0 Oct  5 13:54 c
-rw-r--r--. 1 nobody  nobody  0 Oct  5 18:30 remoteFile
drwxr-xr-x. 2 student student 55 Oct  5 16:31 student
[root@client ~]#
```

4. When you have completed the exercise, you shall comment out the lines that you have added to /etc/fstab so that it would not be automatically mount the LVM nor the NFS filesystem the next time you start your Linux client system.

9. Enable a Shared Folder (Host) for a Virtual Machine (Linux)

This section is very important. It is related to your answer submission during the MST and EST. You need to know how to copy files in between your Linux VM and your host computer (Your notebook) during the MST and EST.

Please watch this [demo video](#) first and try to complete the following exercise:

On client/server

1. Copy the /etc/shadow file to your host system. Store it in **c:/LAS/LAS_share.**
2. Show it to your tutor. (Your will earn GP marks for this.)

Additional Reference:

- An introduction to Linux Access Control Lists (ACLs) - <https://www.redhat.com/sysadmin/linux-access-control-lists>
- Learn to use extended filesystem ACLs - <https://www.techrepublic.com/article/learn-to-use-extended-filesystem-acls/>
- idmapd(8) — Linux manual page - <https://man7.org/linux/man-pages/man8/idmapd.8.html>

End of Practical