**Lesson 3**

**Linux File systems and Network File Service (NFS)**

**Contents**

[1. Setting up an XFS filesystem based on LVM 2](#_Toc84351487)

[2. Filesystem Access Control Lists 11](#_Toc84351488)

[3. Exporting directories on the NFS Server 15](#_Toc84351489)

[4. Mounting exported directories on the NFS Client 17](#_Toc84351490)

[5. NFS Export options 19](#_Toc84351491)

[6. Linux File Permissions vs NFS Permissions 21](#_Toc84351492)

[7. Accessing exported directories by mounting pseudo-root 24](#_Toc84351493)

[8. Mount NFS exported directory on bootup with /etc/fstab 26](#_Toc84351494)

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 is 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 on a huge logical disk storage which is 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.

**Graphical user interface, application

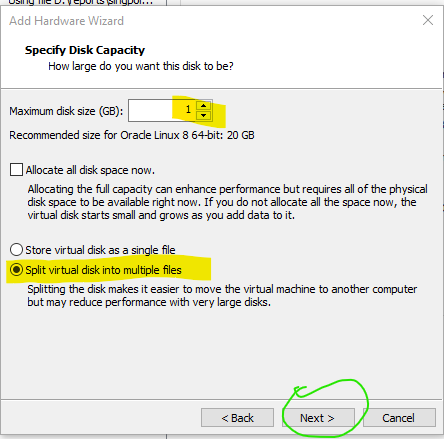
Description automatically generated**

1. In the next screen accept the default disk type (can be SCSI or NVMe - depending on the actual hardware you are having.) , click the Next button. In the next screen accept the default (Create a new virtual disk) option and click the Next button.

Graphical user interface, text, application, email

Description automatically generated

1. 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.



1. Click **Finish** in the next screen to complete the new virtual disk creation.

Now you may start / restart your Client VM.

On client (after it is powered on)

1. Login as root (or 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.)

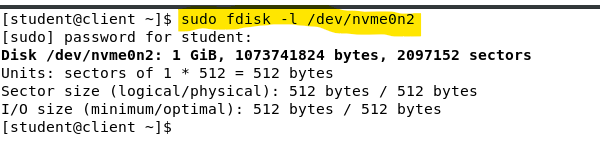
For the following sample listing, the 1 GB new disk is mapped to /dev/nvme0n2.

(You should get a different logical disk name.)

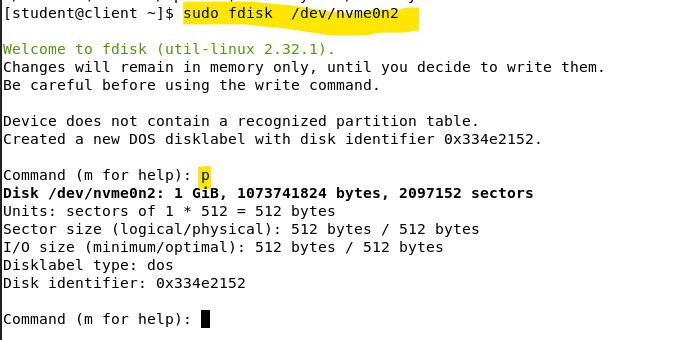


**Viewing and creating new partitions onto the new disk**

1. 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.



1. 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".
2. In fdisk, type 'm' to view the available options.
3. Type “p” to list the existing partitions on the hard disk.



1. To create a new partition, type 'n'.

Follow by typing a 'p' to create a primary partition.

Type "1" for Partition number 1.

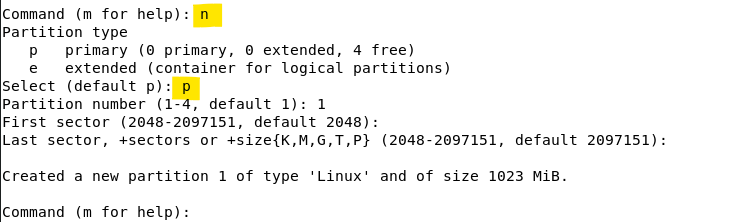
Press enter to accept the default starting sector.

Press Enter

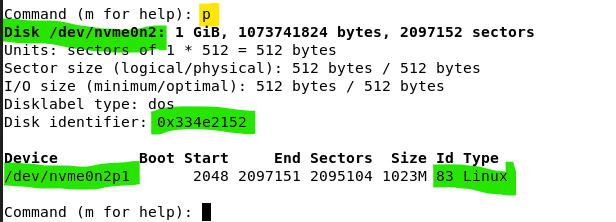
Press enter to accept the default ending sector.

Press Enter

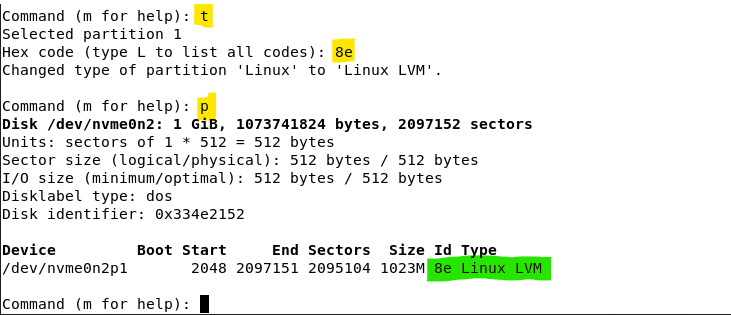
This will use the entire disk to be your partition 1.



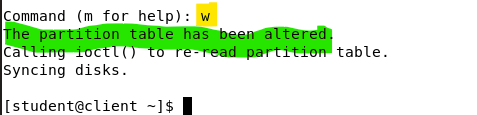
Type 'p' to list the partition info. Note your first partition and its device name (/dev/nvme0n2p1). Also note its Disk identifier, and the Device ID, which is 83 for Linux.



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



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



1. **Restart** your system for the changes in the partition table to take effect.
2. 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:

pvcreate <partition device name>

(Hint: you can run fdisk -l <logical disk device name> to find out the partition device name)

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

vgcreate <vg name> <partition device name>

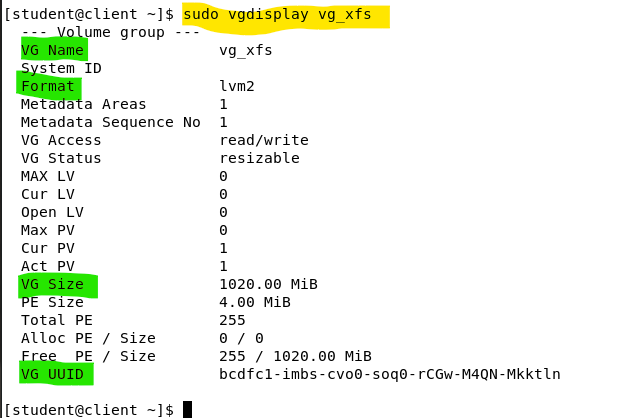
For instance, we will use vg\_xfs as the vg name for this exercise.

Text

Description automatically generated

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

vgdisplay <vg name>



For the above, we have 255 physical extents or 1020 MB free space available in vg\_xfs.

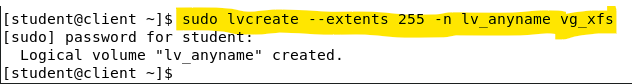
This volume group has been assigned with a unique VG UUID too.

1. Finally, 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 lv\_anyname vg\_xfs

or

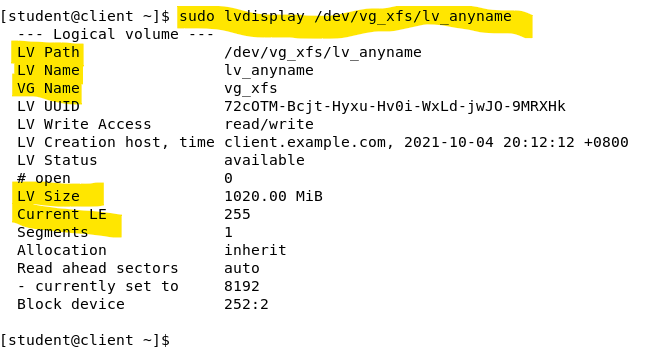
lvcreate --extents 255 -n lv\_anyname vg\_xfs

****

1. 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\_xfs, and the lv name is lv\_anyname:



**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\_xfs/lv\_anyname). Type:

mkfs -t xfs /dev/vg\_xfs/lv\_anyname

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

You may see output like the following :



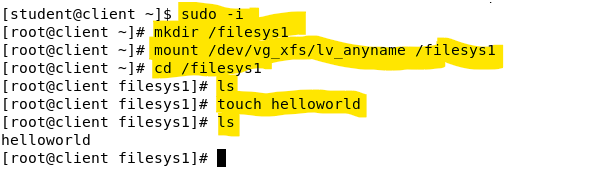
1. As root user, create a mount point and mount the virtual block device. Type

mkdir /filesys1

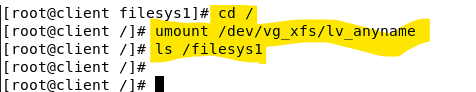
1. Try to mount the filesystem onto /filesys1. Type

mount /dev/vg\_xfs/lv\_anyname /filesys1

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



1. Change to / directory. Unmount the /dev/vg\_xfs/lv\_anyname file system and you will not see the helloworld file in /filesys1 anymore.



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.

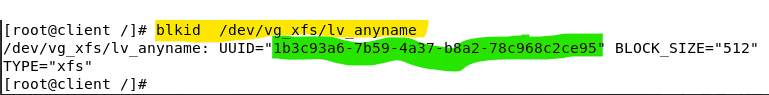
**Mounting an XFS filesystem to the system permanently**

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:

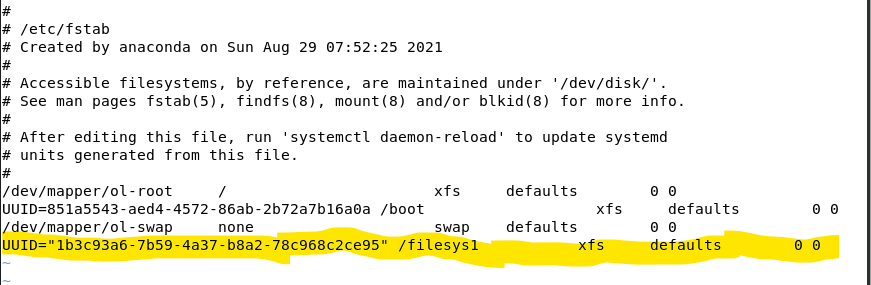
blkid /dev/vg\_xfs/lv\_anyname

For instance:



1. 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=" 6f91ea2b-dc9e-4d04-801a-c647f649e968" /filesys1 xfs defaults 0 0



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

1. 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).

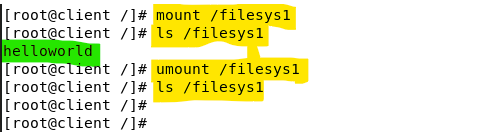
Try:

mount /filesys1

and/or

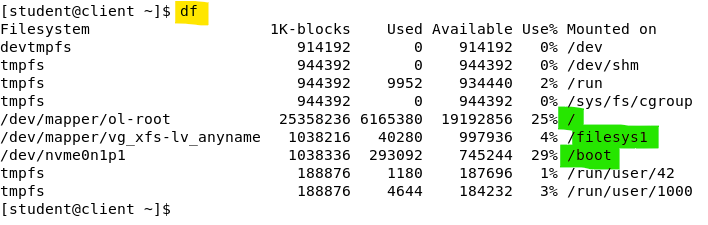
umount /filesys1

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



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

df

****

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 device names for each of the corresponding filesystems.

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 8.4. Both of the LVM and the XFS are set to be used by default.

1. 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 some 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.

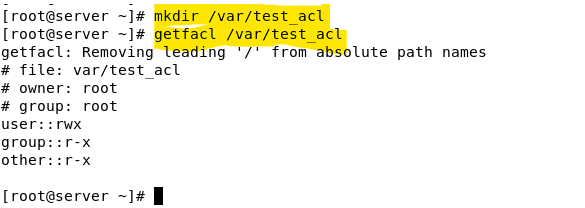
On Server:

1. Login as root and create a folder /var/test\_acl. Type:

mkdir /var/test\_acl

1. Examine the default ACLs for the newly created folder /var/test\_acl. Type:

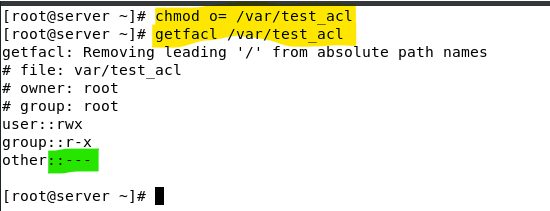
getfacl /var/test\_acl



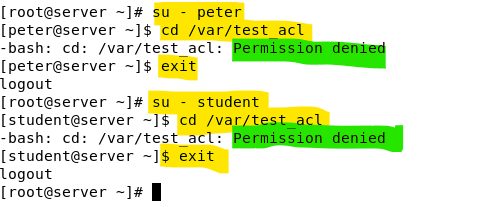
From the above, we learn that the group id of the folder is 'root' and this folder allows any users to read from and access to it.

1. Disable the read and access permission for the other. Type:

chmod o= /var/test\_acl



1. Verify that other users (not in the 'root' group) are not able to access to /var/test\_acl.



For instance, the above shows neither peter nor student can access to the folder /var/test\_acl.

1. Enable the group members of ppm to have read, write and access permission to the /var/test\_acl using ACLs. Type :

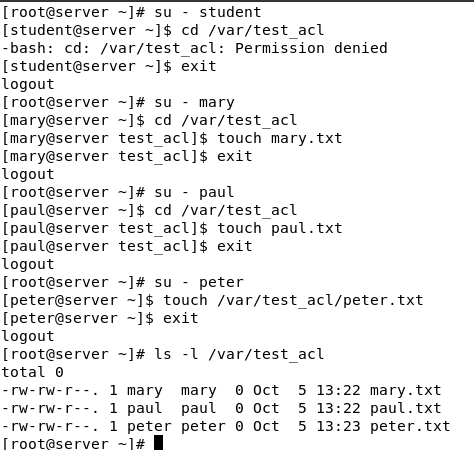
setfacl -m g:ppm:rwx /var/test\_acl

Text

Description automatically generated

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, instead it shows the 'maximum' permission of group in its output. The maximum permission of group is display as the mark field in the output too.

1. 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.



1. Setfacl can also be used to set permission to 'deny' specific users or groups to access to a file/directory.

To deny mary to have the rwx permissons 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.)

Text

Description automatically generated

1. 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.

Text, letter

Description automatically generated

1. 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.

Text, letter

Description automatically generated

1. Exporting directories on the NFS Server

"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

On server:

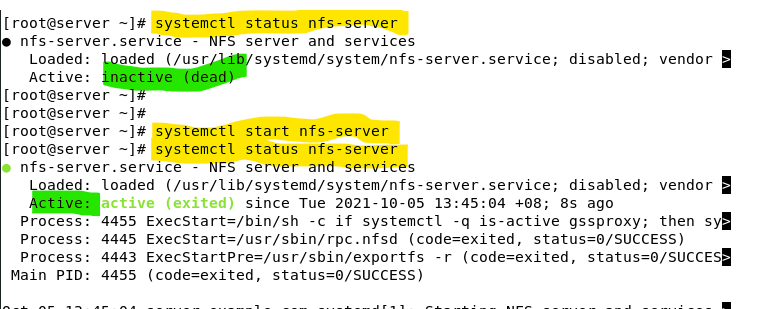
1. Check if the nfs package has been installed:

dnf info nfs-utils

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

systemctl status nfs-server

systemctl start nfs-server



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

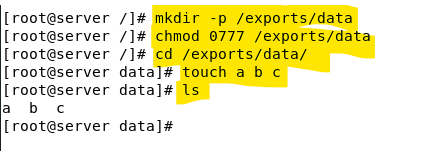
mkdir -p /exports/data

chmod 777 /exports/data

1. Create some files in /exports/data.

cd /exports/data

touch a b c

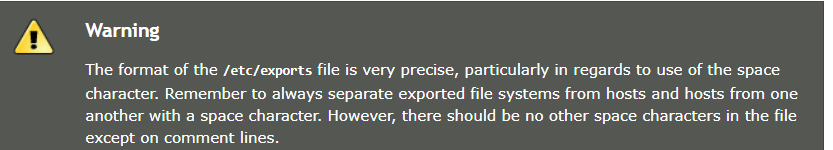


Use -p switch in the mkdir command is to ensure the targeted folder will be created.

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

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

Client ip : 192.168.126.129





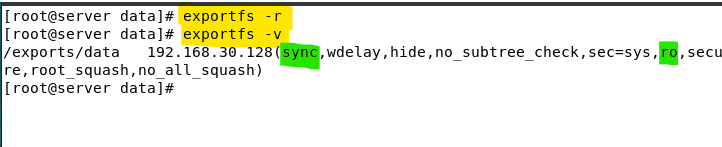
**(My ClientIP is 192.168.30.128)**

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

exportfs -r

1. Run exportfs with -v option to check the exports.

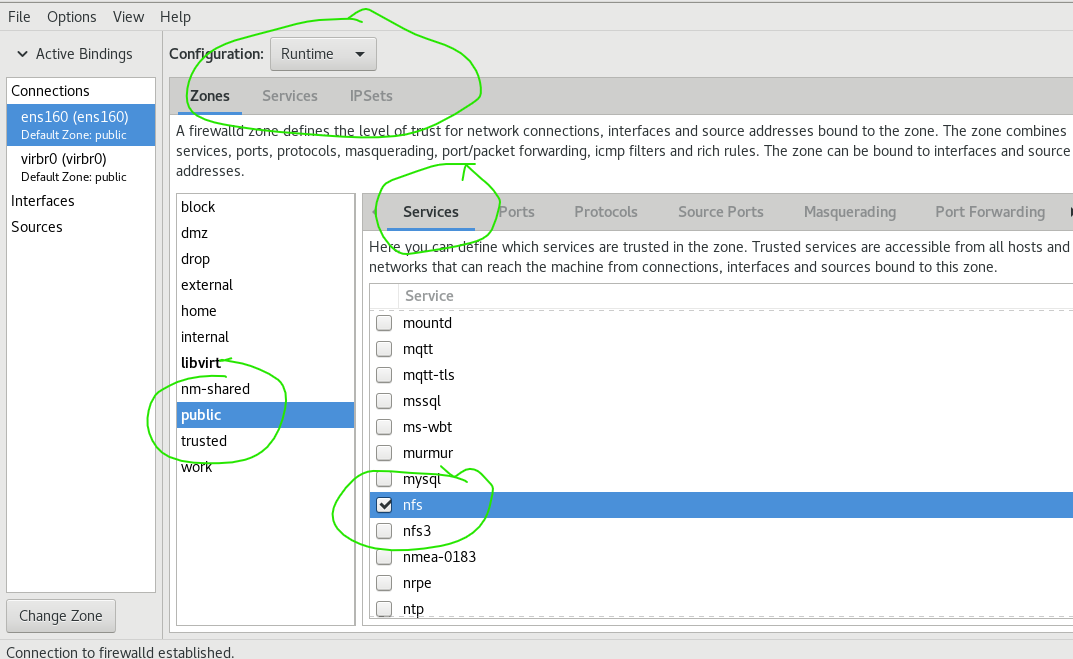
exportfs -v



Take note that, there are many default options have been added to this exported NFS on top the ro (ReadOnly) and sync options we have defined in the /etc/exports configuration file.

1. 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, TCP Port 2049 is sufficient.]

****

1. Mounting exported directories on the NFS Client

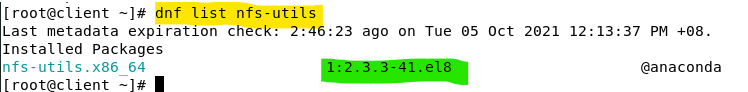
In the previous section, we have be enabled a NFS service at the server to allow the client to access to the /exports/data folder over the network with NFS protocol.

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



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

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

mkdir -p /mount/data

1. Mount\* the exported directory.

mount <*serverIP>*:/exports/data /mount/data –o rw

server ip : 192.168.126.88

\*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.)

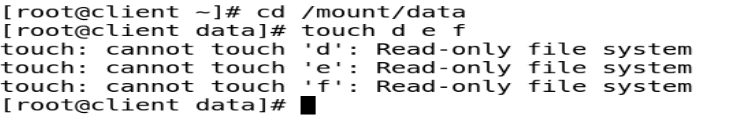
Calendar

Description automatically generated

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.

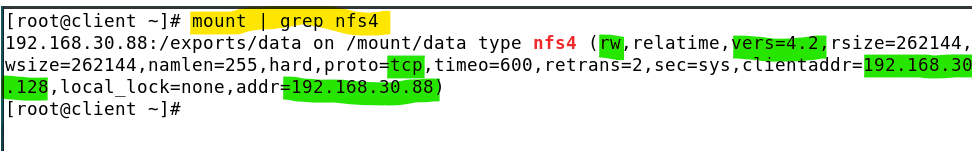
(Note: in case your mount is failed, check the following 2 configurations: Your Server Firewall has openend nfs server (or TCP port 2049), and your server IP is correctly identified.)

1. 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.



1. 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

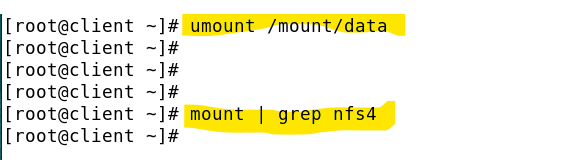


The above show a few interesting and important attributes of the NFS file system. The attribute shows that the file system is rw (read and write) but it is in fact read only.

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 opion.) 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).

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

umount /mount/data



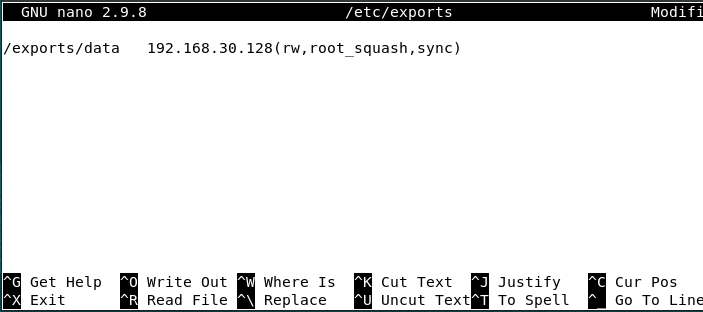
1. NFS Export options

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.

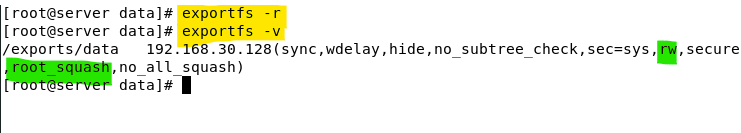
On server:

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

/exports/data *clientIP*(rw,root\_squash,sync)



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



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 uploading of programs with the setuid bit set.

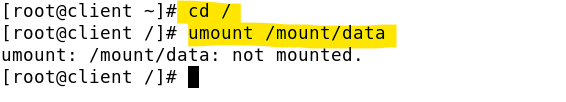
If no\_root\_squash is used, remote root users can change any file on the shared file system and leave applications infected by Trojans for other users to inadvertently execute.

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 :

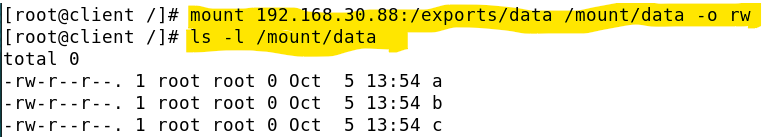
1. Make sure no one is accessing the mounted directory /mount/data.
2. Make sure there is no filesystem mounted using /mount/data as a mount point. Type:

umount /mount/data



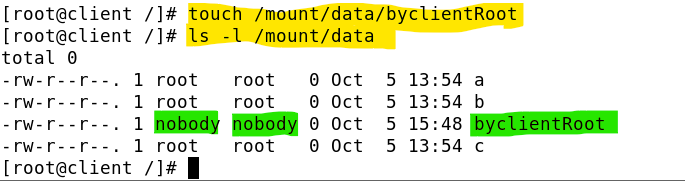
1. Mount the exported directory again.

mount 192.168.126.88:/exports/data /mount/data –o rw



**The mount is successful, and we can see the a, b and c files.**

1. As root, check the contents of /mount/data. Try to create a 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.



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.

On client (login as student):

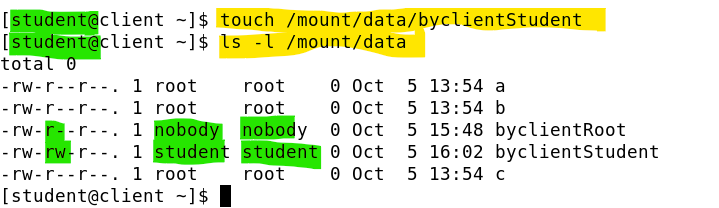
1. As user student, create another file in /mount/data, name it byclientStudent.txt . Type:

touch /mount/data/byclientStudent

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

ls -l /mount/data

Note that files created by root over the NFS share are owned by nobody. This is because by default, directories are exported with the root\_squash option. The user root is mapped to user nobody when accessing the exported directory.



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) and student have different default file permissions.

1. 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. Set the file permissions so only user student can write in this directory.

mkdir -p /exports/data/student

chown student.student /exports/data/student

chmod 755 /exports/data/student

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

Text

Description automatically generated

On client:

1. 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

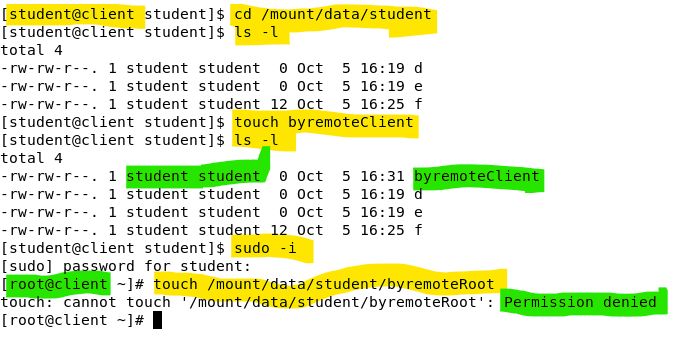
echo hello world >> f

cat f

Text

Description automatically generated

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



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

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

There is a serious issue related to the 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 step, we use the server student to create a file and let the client student to access and update. This operation itself is wrongfully assume these two accounts are used by the same user. In many cases, this is not true.

Note: NFS only use 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.

1. To illustrate the impact of id mapping issue:

On server:

* + - 1. Login in as paul in the server, create a file in the /exports/data directory. Type:

cd /exports/data

touch byPaul

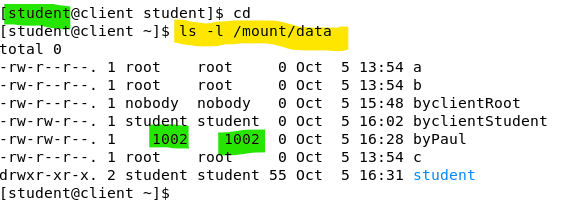
A picture containing table

Description automatically generated

On Client:

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

ls -l /mount/data



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.**

1. Unmount the directory

umount /mount/data

1. Accessing exported directories by mounting pseudo-root

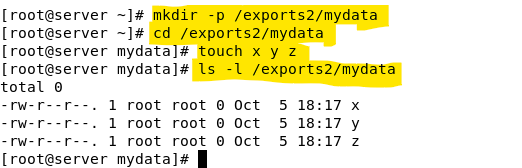
Mounting with pseudo-root can help to simplify the NFS mount operations.

On server:

1. Create another directory /exports2/mydata.

mkdir -p /exports2/mydata

1. Create some files in /exports2/mydata.



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

/exports2/mydata \*(ro,sync)

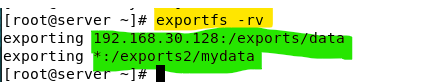
Text

Description automatically generated with medium confidence

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.

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

exportfs -rv



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.

1. Run the following command to mount the pseudo-root of the server using /mount as the mount point. Type:

mount *192.168.126.88*:/ /mount

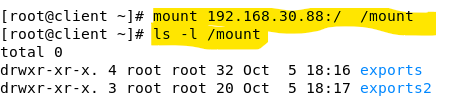
Note: In the previous nfs mount commands used , we specified the actual folder path to be mounted: ie.

mount serverIP:/exports/data /mount/data –o rw

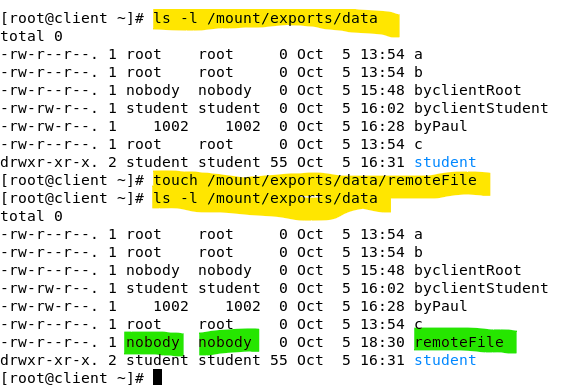
In this exercise, we only use serverIP:/ as the target mount folder (pseudo root).

Do a ls -l of /mount to find out what has been exported from the server.

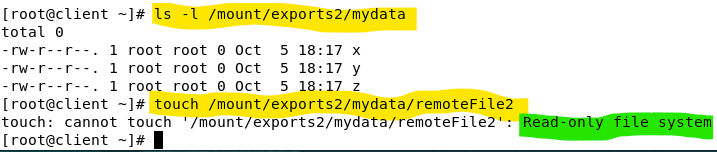
ls -l /mount



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



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



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

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

1. Unmount the directory

umount /mount

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

1. 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.

*192.168.126.88*:/exports/data /mount/data nfs defaults 0 0

Graphical user interface, text

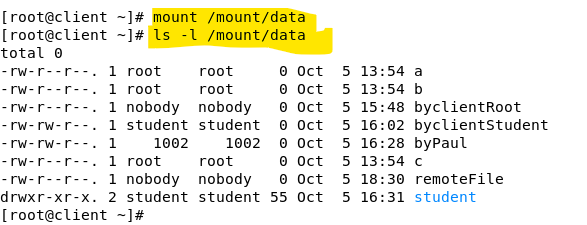
Description automatically generated

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

mount /mount/data

1. Check the contents of /mount/data to see if the server’s exported directory has been mounted.

For instance:



1. 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.

**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*