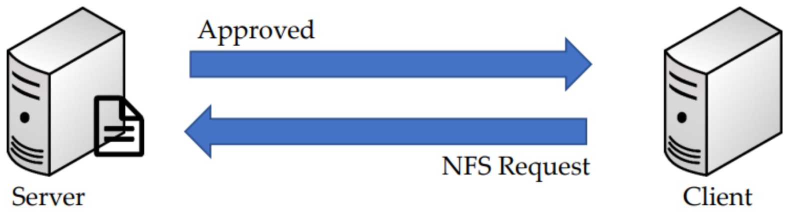
Access network-attached storage (NFS and Samba)

Linux #redhat #network #filesystem

Network File System (NFS)

- This filesystem is not local to your computer.
- Called NAS (Network Attached Storage)
- NFS is one of the protocols on NAS
- NFS stands for Network File System, a file system developed by Sun Microsystems, Inc.
  - This was a company who actually had one of the biggest operating system called Solaris, Sun Microsystems now is being bought by Oracle
- It is a client/server system that allows users to access files across a network and treat them as if they resided in a local file directory
- For example, if you were using a computer linked to a second computer via NFS, you could access files on the second computer as if they resided in a directory on the first computer. This is accomplished through the processes of exporting (the process by which an NFS server provides remote clients with access to its files) and mounting (the process by which client map NFS shared filesystem)



- You have a machine, a computer, a client. This machine wants to attach a filesystem that is being shared from another computer, so it sends an NFS request.
- That NFS request goes to a server that is hosting that filesystem.
- Then based on the rules that are defined in the server, meaning the policies and which server to share with, which server to trust and all that, once that is satisfied, then the server is gonna send the approved request back to its client.

Steps for NFS Server Configuration

- The server will share its filesystem

Install NFS packages

- `yum/dnf install nfs-utils libnfsidmap` (most likely they are installed)
  - Check if already installed with `rpm -qa | grep nfs`

Example using `rpm` command:

```
[root@localhost ~]# rpm -qa | grep nfs
libnfsidmap-2.5.4-25.el9.x86_64
nfs-utils-2.5.4-25.el9.x86_64
sssd-nfs-idmap-2.9.4-6.el9_4.x86_64
```

Enable and start NFS services

- `systemctl enable rpcbind`
- `systemctl enable nfs-server`
- `systemctl start rpcbind, nfs-server, rpc-statd, nfs-idmapd`
  - These are the 4 services that we need to start

Create NFS share directory and assign permissions

- `mkdir /mypretzels` (if you already have a directory that you want to share then you do not need to create a new one)
- `chmod a+rwx /mypretzels`
- Add some files to the directory so it is not empty

Example using `ls` command in the shared filesystem:

```
[root@linuxtest mypretzels]# ls -ltr
total 4
-rw-r--r--. 1 root root  0 Jul  4 17:18 c
-rw-r--r--. 1 root root  0 Jul  4 17:18 b
-rw-r--r--. 1 root root  0 Jul  4 17:18 a
-rw-r--r--. 1 root root 37 Jul  4 17:19 kramer
```

Modify `/etc/exports` file to add new shared filesystem

- The reason we need to modify this file is because if you want to have this filesystem to be shared, we have to have certain option on it
- The following line means we want `/mypretzels` to be shared only with the server that has this IP, and it will have the `rw` permission.
- `/mypretzels 192.168.12.7 (rw,sync,no_root_squash)` = for only 1 host
  - `/mypretzels` = NFS share (this is what you are sharing)
  - `192.168.12.7` = IP address of client machine from which we are gonna share the filesystem with
    - What if you wanted to share with everybody? Then you could simply put a `*` (asterisk) sign instead of an IP.
  - `rw` = Giving permissions to the client
  - `sync` = All changes to the according filesystem are immediately flushed to disk; the respective write operations are being waited for. (don't way any longer when you are writing to this filesystem from a client, just write it to the disk right away)
  - `no_root_squash` = Root on the client machine will have the same level of access to the files on the system as root on the server (client root will have the same permissions as server root)
- `/mypretzels * (rw,sync,no_root_squash)` = for everyone

Create a backup copy
Example using `cp` command:

```
[root@localhost ~]# cp /etc/exports /etc/exports_orig
```

- It is also recommended to put the date in the copy file
- This way if we make a mistake we could always copy it back to its original version.

Edit the `/etc/exports` file
Example using `vi` command:

```
[root@localhost ~]# vi /etc/exports
```

File editor:

```

~
~
~
~
:
```

- Since we are editing `/etc/exports` for the first time you will see the file is empty
- Please be very careful when you modify this file

File editor:

```
/mypretzels     *(rw,sync,no_root_squash)
~
~
~
~
:wq!
```

- First enter what you are exporting (`/mypretzels`)
- Then you can Tab and enter who are you exporting it to (`*` for everyone)
- Then specify permissions and options (`(rw,sync,no_root_sq)`)
- Now save the file by hitting the Esc key and typing `:wq!`

Export the NFS filesystem

- `exportfs -rv`
  - The `-r` option is to republish everything that is inside of `etc/exports`
  - The `-v` option is to show a verbose mode

Example using `exportfs` command:

```
[root@localhost mypretzels]# exportfs -rv
exporting *:/mypretzels
```

- Now it will tell you that it is exporting the filesystem to everybody (`*`), and the filesystem is `/mypretzels` as we defined in the `/etc/exports` file

Make sure `firewalld` or `iptables` stopped (if running)

- `ps -ef | egrep "firewall|iptable"`
- `systemctl stop firewalld`
- `systemctl stop iptables`

Steps for NFS client configuration

- This is the machine that wants to access the shared filesystem
- You can go ahead and create a second Linux Virtual Machine for the sake of experiment.
  - Remember to select to Bridge Network Adapter in the virtual environment (More on Linux System Access (Command Line and GUI))

Install NFS packages

- `yum/dnf install nfs-utils`
  - Check if already installed with `rpm -qa | grep nfs`
- `yum/dnf install rpcbind` (if not already installed)

Enable and start NFS services

- `systemctl enable rpcbind`
  - In comparison to the server side this is the only server that has to be running in the client machine
- You could verify the service is running by doing `ps -ef | grep rpc`

Make sure `firewalld` or `iptables` stopped (if running)

- `ps -ef | egrep "firewall|iptable"`
- `systemctl stop firewalld`
- `systemctl stop iptables`
- Check also on the server side that this services are stopped

Show mount from the NFS server

- `showmount -e 192.168.1.177` (NFS Server IP)
  - Use the `showmount` command to see what is available to see for the client to mount.
  - The `-e` option stands for "exports" and shows the NFS server's export list.

Example using `showmount` command:

```
[root@localhost ~]# showmount -e 192.168.1.177
Export list for 192.168.1.177
/mypretzels *
```

Create a mount point

- `mkdir /mnt/kramer`

Mount the NFS filesystem

- `mount 192.168.1.177:/mypretzels /mnt/kramer`
    - `192.168.1.177` is the IP of the NFS server
    - `/mypretzels` is the shared filesystem
    - `/mnt/kramer` is the mount point we just created

Verify mounted filesystem

- `df -h`

Output from `df-h`:

```
...
192.168.1.18:/mypretzels  8.0G  4.1G  4.0G  51%  /mnt/kramer
```

Going to `/mnt/kramer` and verifying we got the same files from the shared filesystem on the server
Example using `ls` command:

```
[root@localhost ~]# cd /mnt/kramer
[root@localhost kramer]# ls -ltr
total 4
-rw-r--r--. 1 root root  0 Jul  4 17:18 c
-rw-r--r--. 1 root root  0 Jul  4 17:18 b
-rw-r--r--. 1 root root  0 Jul  4 17:18 a
-rw-r--r--. 1 root root 37 Jul  4 17:19 kramer
```

- Note these are the same files that we created on the shared filesystem on the server side.

Modify the filesystem from the client size and verify on the server
Example using `touch` command:

```
[root@localhost kramer]# touch david
[root@localhost kramer]# ls -ltr
total 4
-rw-r--r--. 1 root root  0 Jul  4 17:18 c
-rw-r--r--. 1 root root  0 Jul  4 17:18 b
-rw-r--r--. 1 root root  0 Jul  4 17:18 a
```

```
-rw-r--r--. 1 root root 37 Jul  4 17:19 kramer
-rw-r--r--. 1 root root 37 Jul  4 17:19 david
```

Verify the changes on the server side
Example using `ls` command:

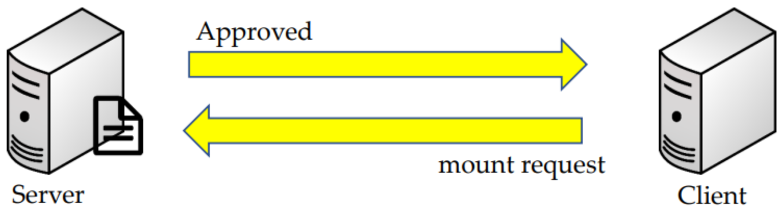
```
[root@localhost mypretzels]# ls -ltr
total 4
-rw-r--r--. 1 root root  0 Jul  4 17:18 c
-rw-r--r--. 1 root root  0 Jul  4 17:18 b
-rw-r--r--. 1 root root  0 Jul  4 17:18 a
-rw-r--r--. 1 root root 37 Jul  4 17:19 kramer
-rw-r--r--. 1 root root 37 Jul  4 17:19 david
```

To unmount

- `umount /mnt/kramer`
- You have to get out of the directory in order to unmount the filesystem

Samba

- Samba is a Linux tool or utility that allows sharing for Linux resources such as files and printers to with other operating systems
- It works exactly like NFS but the difference is NFS shares within Linux or Unix like system whereas Samba shares with other OS (e.g. Windows, MAC etc.)
- For example, computer "A" shares its filesystem with computer "B" using Samba then computer "B" will see that shared filesystem as if it is mounted as the local filesystem



Samba (smb vs. CIFS)

- Samba shares its filesystem through a protocol called **SMB** (Server Message Block) which was invented by IBM
- Another protocol used to share Samba is through **CIFS** (Common Internet File System) invented by Microsoft and NMB (NetBios Named Server)
- CIFS became the extension of SMB and now Microsoft has introduced newer version of SMB v2 and v3 that are mostly used in the industry
- In simple term, most people, when they use either **SMB** or **CIFS**, are talking about the same exact thing

Samba Installation and Configuration

- Take snapshot of your VM
- Install samba packages
- Enable samba to be allowed through firewall (Only if you have firewall running)
- Disable firewall
    - Please get the permission to disable the firewall service if you are in production
- Create Samba share directory and assign permissions
- Change SELinux security context for the samba shared directory
- Or disable SELinux
    - Please confirm with your security team if in a production environment
- Modify `/etc/samba/smb.conf` file to add new shared filesystem
    - Please make a backup copy of the `smb.conf` file
- Verify the setting
- Once the packages are installed, enable and start **Samba** services (smb and nmb)
- Mount Samba share on Windows client
- Mount Samba share on Linux client using CIFS
- Create secure Samba share

Samba step by step installation instructions

- In this note we will be doing exercises with Samba to know how it works so we will need different machines to test this:
    - The host 'linuxtest' will be our Samba server this time
    - Then we have a Windows machine from which we will access the Samba share
    - Finally a second Linux machine this time for client will be 'monks'
- First please make sure to take a snapshot of your VM

Install samba packages

- Become root user
- `yum/dnf install samba samba-client samba-common`
- Run `rpm -qa | grep samba` to verify that the packages were installed

Enable samba to be allowed through firewall (Only if you have firewall running)

- `firewall-cmd --permanent --zone=public --add-service=samba`
- `firewall-cmd -reload`

To stop and disable `firewalld` or `iptables`

- `systemctl stop firewalld`
- `systemctl stop iptables`
- `systemctl disable firewalld`
- `systemctl disable iptables`

Create a Samba share directory and assign permissions

- `mkdir -p /samba/morepretzels`
  - The `-p` option stands for "parents" and specifies to no error if existing, make parent directories as needed
  - You could pick any name for this directory
- `chmod a+rwx /samba/morepretzels`
  - Now everyone could read, write, and execute
- `chown -R nobody:nobody /samba`
  - Changing the ownership
  - If you originally go to `/` and do `ls -l`, you will notice that the `/samba` directory is owned by root (since we created it while on root user). We want to change its ownership permissions to nobody.
  - The `-R` options stands for "recursive" and specifies to cascade down the action in that directory
  - Note we are specifying `nobody` for user then `:` and then `nobody` for the group as well

Example using `ls` command:

```
[root@localhost ~]# ls -ltr /
```

Output:

```
...
drwxr-xr-x.   3 nobody nobody   26 Jul  4 19:10 samba
....
```

- The ownership changed from root to nobody
- Confirm the same with the contents of the `/samba` directory

Change the SELinux security context for the samba shared

- `sestatus` (To check the SELinux status)
- `vi /etc/selinux/config`

Output from `sestatus`:

```
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      33
```

- It is enabled and enforcing so we want to disable it
- If you are running in production please check with your security folks before disabling SELinux

Example using `vi` command:

```
[root@localhost ~]# vi /etc/selinux/config
```

File editor:

```


# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
#SELINUX=enforcing
SELINUX=disabled
...
```

- Uncomment or add the line `SELINUX=disabled` to disable SELinux completely
- After saving the file do a `reboot` to reload the config file.
- After rebooting when you run `sestatus` command you will see the following lien

Output from `sestatus`:

```
SELinux status:                 disabled
```

Modify `/etc/samba/smb.conf` file to add new shared filesystem

- (Make sure to create a copy of `smb.conf` file)

Example using `cp` command:

```
[root@linuxtest ~]# cd /etc/samba
[root@linuxtest samba]# ls -ltr
total 20
-rw-r--r--. 1 root root 11319 May  1 13:13 smb.conf.example
-rw-r--r--. 1 root root   853 May  1 13:13 smb.conf
-rw-r--r--. 1 root root    20 May  1 13:13 lmhosts
[root@linuxtest samba]# cp smb.conf smb.conf.orig
[root@linuxtest samba]#
```

Modifying the `smb.conf` file
Example using `vi` command:

```
[root@linuxtest samba]# vi smb.conf
```

File editor:

```
# See smb.conf.example for a more detailed config file or
# read the smb.conf manpage.
# Run 'testparm' to verify the config is correct after
# you modified it.
```

```
#
# Note:
# SMB1 is disabled by default. This means clients without support for SMB2 or
# SMB3 are no longer able to connect to smbd (by default).

[global]
        workgroup = SAMBA
        security = user

        passdb backend = tdbsam

        printing = cups
        printcap name = cups
        load printers = yes
        cups options = raw

[homes]
        comment = Home Directories
        valid users = %S, %D%w%S
        browseable = No
        read only = No
        inherit acls = Yes

[printers]
        comment = All Printers
        path = /var/tmp
        printable = Yes
        create mask = 0600
        browseable = No

[print$]
        comment = Printer Drivers
        path = /var/lib/samba/drivers
        write list = @printadmin root
        force group = @printadmin
        create mask = 0664
        directory mask = 0775
```

- This is how the default file looks like
- Go ahead and delete everything below the comment section
  - Put your cursor on the first line where it says `[global]` and just keep pressing the 'D' key to delete everything that's below

Delete everything from the `smb.conf` file and add the following parameters
File editor:

```
[global]
                workgroup = WORKGROUP
                netbios name = centos
                security = user
                map to guest = bad user
                dns proxy = no

[Anonymous]
                path = /samba/morepretzels
                browsable = yes
                writable = yes
                guest ok = yes
```

```
                   guest only = yes
                   read only = no
```

- This is how the file should look like after modifying it
- These are just some default settings
- `[Anonymous]` is going to be the actual share that's gonna show up, and the path is gonna be `/samba/morepretzels`, this is the filesystem that we created to be share
- All the followed permissions are set there
  - Everybody could use it, everybody could browse it, etc.
- Escape and save out of it.

Verify the setting

- `testparm`
  - This command will exactly going to test all the parameters that we have for Samba

Example using `testparm` command:

```
[root@linuxtest ~]# testparm
```

Output:

```
Load smb config files from /etc/samba/smb.conf
Loaded services file OK.
Weak crypto is allowed by GnuTLS (e.g. NTLM as a compatibility fallback)

Server role: ROLE_STANDALONE

Press enter to see a dump of your service definitions

# Global parameters
[global]
        dns proxy = No
        map to guest = Bad User
        netbios name = CENTOS
        security = USER
        idmap config * : backend = tdb


[Anonymous]
        guest ok = Yes
        guest only = Yes
        path = /samba/morepretzels
        read only = No
```

- It will ask you to hit Enter
- If no errors are coming up, everything is perfectly set up until here.

Enable and start Samba services

- `systemctl enable smb`

- `systemctl enable nmb`
- `systemctl start smb`
- `systemctl start nmb`
  - Remember that the reason we enable the service is to let the service start on boot
  - Always check the status of the services to be sure.

Mount on Windows client

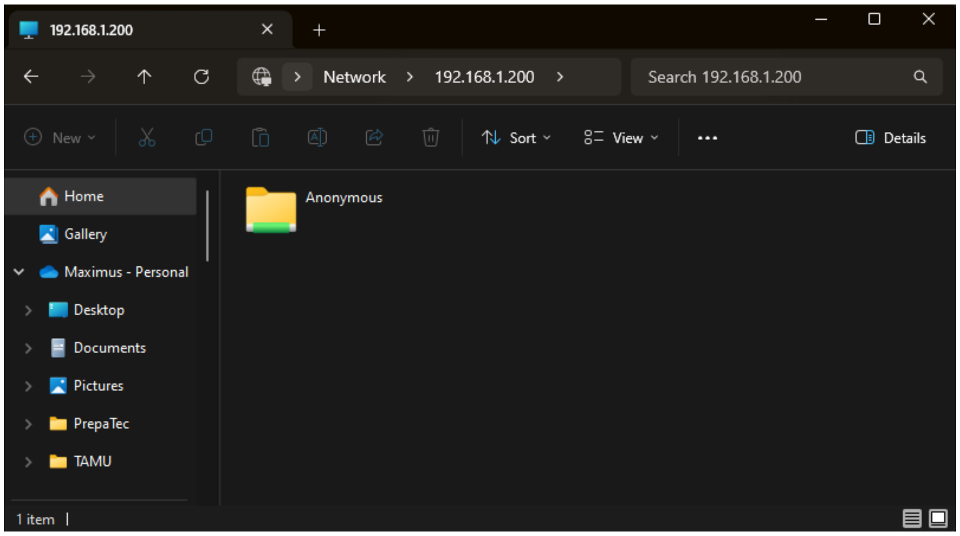
- First create any file to later verify the contents of the share

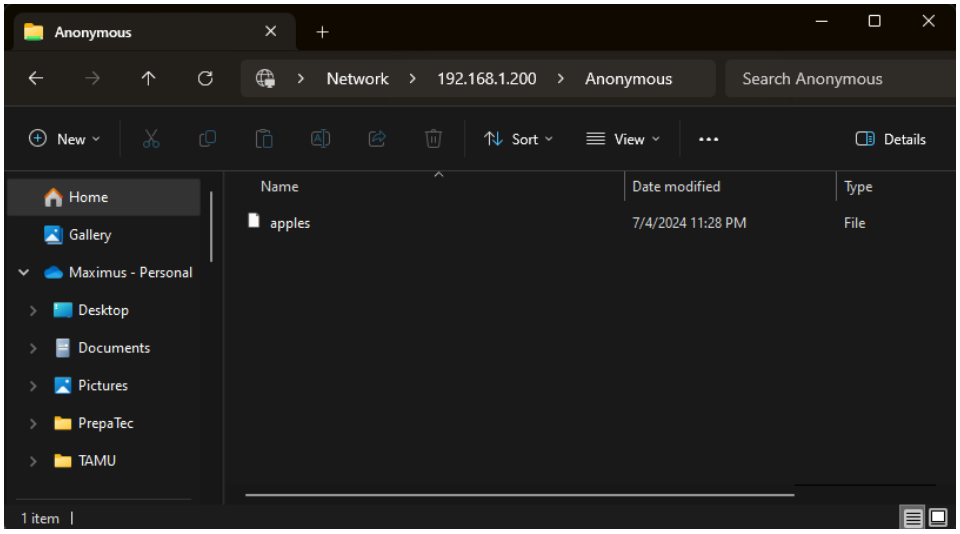
Example using `touch` command:

```
[root@linuxtest samba]# cd /samba/morepretzels/
[root@linuxtest morepretzels]# ls -ltr
total 0
[root@linuxtest morepretzels]# touch apples
[root@linuxtest morepretzels]# ls -ltr
total 0
-rw-r--r-- 1 root root 0 Jul  4 23:28 apples
[root@linuxtest morepretzels]#
```

Now on your windows machine:

- Open File Explorer
- At the top select the path box and type the following
  - `\\192.168.1.200`
    - This is the IP of your server



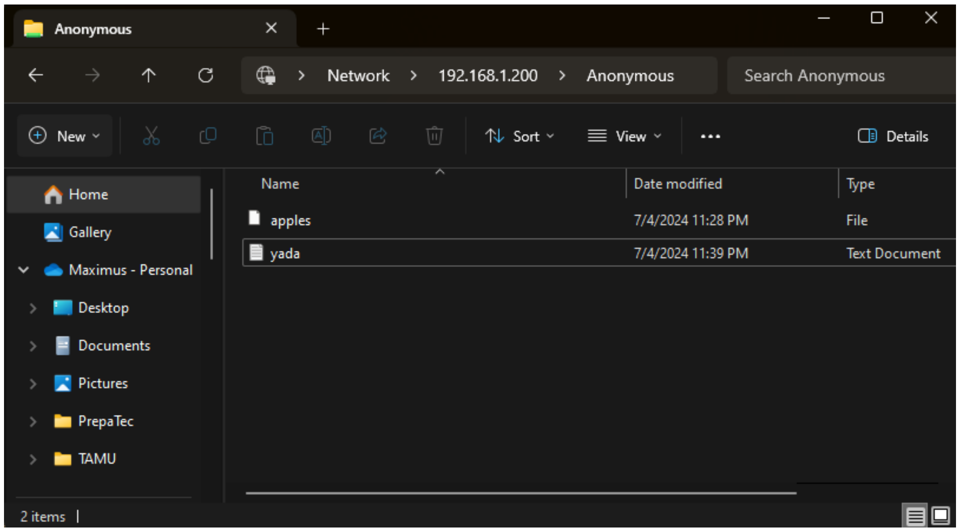


Name	Date modified	Type
apples	7/4/2024 11:28 PM	File

- 'apples' is the file that we just created on the server side.
- This is the same file that you have shared from the server

Go ahead and create a file from the Windows machine

- This way you would know it is showing up on the Linux machine as well.
- Go ahead and:
  - right-click -> New -> Text document



Name	Date modified	Type
apples	7/4/2024 11:28 PM	File
yada	7/4/2024 11:39 PM	Text Document

Now go back to the Linux machine and verify
Example using `ls` command:

```
[root@linuxtest morepretzels]# ls -ltr
total 0
-rw-r--r-- 1 root   root   0 Jul  4 23:28 apples
-rwxr--r-- 1 nobody nobody 0 Jul  4 23:39 yada.txt
```

- Now you will see the `yada.txt` file on here as well
- Note apples was created by root because we created that file while we were logged in as root user

Mount on Linux client

- Become root
- `yum/dnf -y install cifs-utils samba-client`
- Create a mount point directory
  - `mkdir /mnt/sambashare`
- Mount the samba share
  - `mount -t cifs //192.168.1.200/Anonymous /mnt/sambashare/`
    - The `-t` option is for which filesystem (The argument following the -t is used to indicate the filesystem type.)
    - Note `192.168.1.200` is the IP address of the server machine and we are indicating to access the share `Anonymous` filesystem that we created in the `smb.conf` file. Finally we specify the mountpoint, where are we mounting it on?
    - Once you hit Enter it will prompt you for a password. WHAT?
- Entry without password
  - On this point just simply go ahead and hit Enter, do not type any password or don't put any keys

Verify the share is mounted
Example using `df -h` command:

```
[root@monks ~]# df -h
```

- Note that in this example the client hostname is 'monks'

Output:

```
...
//192.168.1.200/Anonymous  8.0G  5.1G  3.0G  63%  /mnt/sambashare
```

Verify we can access the files that we previously created on the share
Example using `ls` command:

```
[root@monks ~]# cd /mnt/sambashare/
[root@monks sambashare]# ls -l
-rw-r--r-- 1 root   root   0 Jul  4 23:28 apples
-rwxr--r-- 1 nobody nobody 0 Jul  4 23:39 yada.txt
```

- Note these are the same files we could access on Windows, as well as on the Linux server.
- You could also create another file here and you can see it both on the Linux server and on the Windows client.

Extra: Secure Samba Server

Create a group smbgrp & user larry to access the samba server with proper authentication

- `useradd larry`
- `groupadd smbgrp`
- `usermod -a -G smbgrp larry`
- `smbpasswd -a larry`

Example using `smbpasswd` command:

```
[root@linuxtest ~]# smbpasswd -a larry
```

Output:

```
New SMB password: YOUR SAMBA PASS
Retype new SMB password: REPEAT YOUR SAMBA PASS]
Added user larry
```

Create a new share, set the permission on the share

- `mkdir /samba/securepretzels`
- `chown -R larry:smbgrp /samba/securepretzels`
- `chmod -R 0770 /samba/securepretzels`
- `chcon -t samba_share_t /samba/securepretzels`
    - This las line is to change the SELinux security context (label).

Edit the configuration file `/etc/samba/smb.conf`

(Create a backup copy first)

- `vi /etc/samba/smb.conf`
    - Add the following lines

File editor:

```
[Secure]
                path = /samba/securepretzels
                valid users = @smbgrp
                guest ok = no
                writable = yes
                browsable = yes
```

File editor:

```
# See smb.conf.example for a more detailed config file or
# read the smb.conf manpage.
# Run 'testparm' to verify the config is correct after
# you modified it.
#
# Note:
# SMB1 is disabled by default. This means clients without support for SMB2 or
# SMB3 are no longer able to connect to smbd (by default).

[global]
                workgroup = WORKGROUP
                netbios name = centos
                security = user
                map to guest = bad user
                dns proxy = no
```

```
[Anonymous]
                path = /samba/morepretzels
                browsable = yes
                writable = yes
                guest ok = yes
                guest only = yes
                read only = no

[Secure]
                path = /samba/securepretzels
                valid users = @smbgrp
                guest ok = no
                writable = yes
                browsable = yes
```

- The `smb.conf` file should now look something like this
- Note that now we have two shared filesystems, one is public and the other is private and is only reachable by members of the `smbgrp`

Restart the services

- `systemctl restart smb`
- `systemctl restart nmb`