NFS- Network File System
- Originally developed by Sun, access to remote file systems with mount points. NFSv4 listens on port 2049

	NFSv3	**NFSv4**
Packages	nfs-utils and nfs-utils-lib	nfs-common, nfs4-acl, rpc-svc-gss, for client. For server, use nfs-kernel-server and nfs-utils
Config files	/etc/sysconfig/nfs, /etc/exports	/etc/exports, /etc/nfs/, /etc/nfs/nfs.conf (server)
Services (SystemD & init)	/etc/init.d/nfs, /etc/init.d/rpcbind service nfs start, service rpcbind start	nfs-server.service and nfs-kernel-server.service, rpc-svc-gss.service
Permissions	/etc/exports file, some options given	ACLs and more options added in /etc/exports
RPC	rpc.lockd, rpc.statd (file locking, status info)	Now automated/ dynamic in NFS4 protocol
Automount	/etc/fstab/	/etc/fstab/ and autofs

NFSv3 Common permission options in the /etc/exports file
Self-explanitory: read-write (rw), read-only (ro), and no access (-)

root_squash	Forces all NFS requests from the root user (UID 0) on the client machine to be mapped to the anonymous user (usually nfsnobody) on the server, to help prevent accidental or malicious modifications by the root user on the client. [also "chown nfsnobody /share" for shared folder]
no_root_squash	Provides access with full permissions (if granted in the ACL). Use this option with caution due to potential security risks.
all_squash	A more aggressive version of root_squash that maps all client UIDs to the anonymous user on the server, regardless of the client user (has significant security implications).
anonuid/anongid	Allow you to specify the UID and GID of the anonymous user on the server used for mapping client users when root_squash or all_squash is enabled.
sync/ async	These have the same functionality as used with the mount command

You can specify access rights for specific hosts or networks. Wildcards can be used to represent groups of hosts or networks in the access control list. you can also use CIDR notation like 192.168.1.0/24

NFSv4-introduced new features and functionalities
Added options for overriding inheritance and setting specific permissions for individual directories in /etc/exports
- Fake root mount: if a server exporting /home and /data, instead of mounting both, just mount /
- Allows for clients to send their UID/GID info for access control decisions
- Access Control Lists (ACLs) on exported directories- the nfs4-acl package provides extensions for setfacl and getfacl to better support NFS, for more granular access permissions for specific users and groups on the server.
- Security options provide more granular authentication and authorization mechanisms like using a Kerberos server. An older option NLM server (Network Lock Manager) is supported but outdated, and NFS delegation tickets (built into NFS) has been proved to have vulnerabilities, thus Kerberos solutions seem to remain as the best security solution. Kerberos options are krb5p, krb5i, and none. Use krb5p for Kerberos security with encryption for data privacy and integrity protection. The krb5pi uses Kerberos but without data encryption. None is obviously not recommended. There was also a systemd sevrvice called nfs-secure-server.service but it integrated into nfs-server.service

Some sample entries in the NFSv4 /etc/exports file
```
/data *(rw,sync,all_squash)
        # Export /data directory with read-write access for all (only all_squash for basic security- this is weak and not advised)
/directory_to_export -sec=krb5p,rw,root_squash,sync  # Consider using ACLs instead of root_squash
        # Export with Kerberos, read-write access, root squashing, and sync
/home/users (rw,sync,nfsv4,user_acl,sec=krb5p)  # Preferred approach
        # Export /home/users with user ACLs, NFSv4, and Kerberos security
```

Commands:
Use exportfs to manage NFS exports. Options include -v to list currently exported directories; -a to export all entries in /etc/exports. To add new exports use 'exportfs -o options /dir client_IP' and to remove exports use 'exportfs -u /directory'
The command 'showmount -e' gives info on NFS shares currently exported; rpcinfo nfs gives info about the NFS server's RPC, and nfsstat gives statistics related to NFS server activity.

Client-side: mount, showmount -e server_IP (show NFS shares on a server), and nfsstat for NFS client statistics

The autofs package:
autofs.service: The systemd service that manages the autofs daemon itself. (start, stop, and restart)
automount: Command for manual interaction with autofs, manually mounting or unmounting specific automount points or debugging automount behavior (viewing logs or checking status of automount points)

/etc/auto.master: main configuration file - defines mount points, links to their map files, global options for autofs
Each line in the file typically follows this format: <mount_point> <map_file> [<options>]
/etc/auto.<identifier>: Map files, referenced by the master map file, contain details on individual automount points.
Each map file defines a specific automount configuration for a particular mount point. The format depends on the chosen map type:
amd.map format: Traditional format that specifies the NFS server location, automount behavior, access control.
auto.master.d format: Newer format that allows for inheritance and modular configuration based on directories.

NFSv3 Mount Options:
rsize/ wsize (read/write size): Set maximum packet size per request the client will try to read or write from the NFS server. Increasing can improve performance for large file reads, but values too large might lead to fragmentation and inefficient network usage.

bg/fg (background/foreground): background allows the system to continue booting while the NFS mount is being established, while foreground forces the mount command to wait until the NFS mount is successfully completed before returning.

async/ sync: The default, asynch allows the client to acknowledge write requests to the server before data is physically written to disk. Can improve performance but might lead to data loss if the client crashes before the write is completed. Using sync ensures that all data written to the NFS share is flushed to the server's disk before the mount command returns. This can improve data integrity but can also impact performance.

noauto: prevents the system from automounting the NFS share during boot

_netdev: Tells system to wait for the network interface to be configured before attempting the NFS mount

tcp/udp: NFSv3 typically defaults to TCP, but UDP can be used in specific scenarios (like low-latency networks) with trade-offs in reliability.

NFSv4 Mount Options:
Building on NFSv3 options, NFSv4 introduces additional options related to security flavors and performance optimizations:

sec: krb5p/ krb5i/ none: Use krb5p for Kerberos security with encryption for data privacy and integrity protection. The krb5pi uses Kerberos but without data encryption. None is obviously not recommended. Kerberos needs to be separately configured on the client and server. rpc_sec is a remnant of NFSv2 and no longer relevant.

nfsvers (NFS version): While the system might negotiate the NFS version with the server, you can explicitly specify nfsvers=4 to force an NFSv4 mount.

minor_version (minor NFS version): This option allows specifying a specific minor version of NFSv4 if your server supports multiple versions.

readdirplus: This option enables the client to request additional information along with directory listings, potentially improving performance for browsing directories on the NFS share.

Many NFSv3 mount options like rsize, wsize, sync/async, noauto, background/foreground and _netdev are still relevant for performance tuning and basic mount behavior in NFSv4.

Kerberos setup

Log in to the kerberos server	ssh <username>@<kerberos_server_ip>
Launch kadmin.local to enter CLI	sudo kadmin.local
Add the service principal	addprinc -randkey nfs/<nfs_server_hostname>
Create a keytab file for NFS Server	ktadd -k /etc/krb5.keytab nfs/<nfs_hostname> (hit return, type quit to exit CLI)
Copy file to NFS Server	scp /etc/krb5.keytab <username>@<nfs_server_ip>:/etc/krb5.keytab (hit return, then exit ssh)
Set write permissions for keytab file	sudo chmod 400 /etc/krb5.keytab; sudo chown root:root /etc/krb5.keytab
Be sure /etc/exports file has shares fixed	sec=krb5p or similar, as demonstrated before
Edit /etc/sysconfig/nfs (enable GSS/Kerb)	sudo vi /etc/sysconfig/nfs Add or uncomment lines RPCGSSDARGS="" and RPCSVCGSSDARGS=""
Edit /etc/krb5.conf to check config	sudo vi /etc/krb5.conf - Ensure [realms], [domain_realm] are configured
Enable, start NFS and kerberos	sudo systemctl enable nfs-server rpcbind && sudo systemctl start nfs-server rpcbind && sudo systemctl restart nfs-server
Run on client to verify it's working	sudo mount -t nfs -o sec=krb5p $server_hostname:/$share_name /mnt/nfs

firewalld configuration: sudo firewall-cmd --permanent --add-service=nfs
iptables configuration: sudo iptables -I INPUT -p tcp --dport 2049,111 -j ACCEPT
List SELinux stuff with semanage boolean -l | grep -i '(nfs_)'; semanage fcontext -l | grep -i '(nfs_)'

Samba - SMB

Client executables (install package: samba-client)

- smbcontrol — Manage Samba shares, view connections to servers), and perform client administrative tasks
- smbclient — Browse, copy, manage files/directories on remote Samba servers.
- smbmount — Mount remote Samba shares as local directories.
- nmblookup — Perform NetBIOS name resolution (find Samba servers on network).
- smbcacls — Remote management tool- view and modify Windows ACLs on files hosted on a Samba server.

SystemD services on the client

- smbd.service — Main SMB/CIFS daemon, handles file/print services.
- nmbd.service — Provides NetBIOS name service.
- cifs.service — Mounts CIFS/SMB shares

Server executables (install package: samba)

- samba-tool — For managing configuration, users, shares, Kerberos settings, and other administrative tasks
- smbstatus — Shows the current status of the Samba server, including active connections and shares.
- testparm — Verifies the syntax of your /etc/samba/smb.conf file before restarting the service.
- smbpasswd — Changes Samba user account passwords.

SystemD services on the server (also includes the client list)

- smbd.service — Main SMB/CIFS daemon, providing access to clients, handles file/print services.
- nmbd.service — Provides NetBIOS name service.
- samba.service — Alias for smbd.service.
- winbind.service — Allows Windows domain authentication.
- samba-ad-dc.service — Samba Active Directory Domain Controller service

Main config file is /etc/samba/smb.conf (resources, user authentication, security, etc.)
Optional configs: /etc/samba/smb-security.conf (security settings) or /etc/samba/smb.secrets (for sensitive info)

/etc/samba/smb.conf - specify shared directories or files; access permissions for users and/or groups; LDAP and encryption options, and security settings

- Shares: Define shared directories or files using the [sharename] section.
- Permissions: read only = yes, writeable = yes, and specific user/group entries.
- LDAP Integration: security = ads option and specifying LDAP server details.
- Security Settings: Enhance security with options like:
  - encrypt passwords = yes - Encrypts user passwords during storage.
  - map to guest = bad user - Disables guest access.
  - browsable = no - Hides the share from browse lists.
  - valid users = @users - Restricts access to the local usernames or users in group specified.

Example smb.conf:

```
[Global]
  workgroup = MYWORKGROUP  # Name of your workgroup for network browsing
  server string = My Samba Server  # Descriptive name for your Samba server
  security = ads  # Enable LDAP security for user authentication
  encrypt passwords = yes  # Encrypt user passwords for added security
  map to guest = bad user # No access for the unauthenticated- gives guest access as 'bad user' which doesn't exist (denied!)
  # wins server = 10.0.0.1  # WINS server IP for name resolution, logging, and specifying one interface to listen (not all)
  logging = file  # Enable file-based logging
  log level = warn  # Log warnings and more severe messages
  log file = /var/log/samba/smb.log  # Specify the log file location
  # interfaces = 192.168.1.0/24  # Example: Listen only on the 192.168.1.0/24 subnet
[SharedFolder]
  path = /home/share  # Path to the directory you want to share
  browsable = no  # Hide this share from network browsing
  writable = yes  # Allow users with access to modify files
  read only = no  # Allow both reading and writing
  valid users = @share_access  # Grant access only to users in the "share_access" group
  create mask = 0664  # Example: New files get rw-rw---- permissions (user:group:others)
  directory mask = 0775  # Example: New directories get rwxrwxr-x permissions
[AnotherShare]
  path = /var/www/html  # Path to the web server's document root (example)
  read only = yes  # Allow users to only read files in this share
  valid users = user1 user2 @web_admins  # Grant access to specific users and a group
  # locking = ...  # Options for controlling how multiple clients share access to files
  # oplocks = ...  # Options related to optimistic locking (advanced)
  # write cache (or 'read cache') = yes  # Enable caching writes/ reads for faster performance
```

The "map to guest = bad user" matches any user that fails authentication (denied)

Kerberos implementation

Log in to the Kerberos server	ssh <username>@<kerberos_server_ip>
Launch kadmin.local to enter CLI	sudo kadmin.local
Add the service principal	addprinc -randkey smb/<smb_server_hostname>
Create a keytab file for SMB server	ktadd -k /etc/krb5.keytab smb/<smb_hostname> (hit return, type quit to exit CLI)
Copy file to SMB Server	scp /etc/krb5.keytab <username>@<smb_server_ip>:/etc/krb5.keytab (hit return, then exit ssh)
Set write permissions for keytab file	sudo chmod 400 /etc/krb5.keytab; sudo chown root:root /etc/krb5.keytab
Edit /etc/krb5.conf to check config	sudo vi /etc/krb5.conf - Ensure [realms], [domain_realm] are configured
Ensure /etc/samba/krb5users is ready	The file will need entries similar to example provided below
Enable, start SMB and kerberos	sudo systemctl enable smb rpcbind && sudo systemctl restart smb
Run on client to verify it's working	sudo mount -t cifs -o sec=krb5i,username=<client_username>@<REALM> //$server_hostname/$share_name /mnt/smbshare

Create a user mapping file to translate Kerberos principals (username@REALM) to specific Samba usernames

```
# /etc/samba/krb5users
# Map Kerberos principal "user1@MYREALM" to Samba user "samba_user1"
user1@MYREALM = samba_user1   # samba_user1 would be the username on the Linux host
```

```
[Global]
  workgroup = MYWORKGROUP
  server string = My Samba Server
  security = krb5i  # Enable Kerberos integration with user mapping
  encrypt passwords = yes
  map to guest = bad user
  username map = /etc/samba/krb5users  # Specify the user mapping file location
[SharedFolder]
  path = /home/share
  browsable = no
  writable = yes
  read only = no
  # Allow access only to users mapped in the user mapping file
#  valid users = %    using % means all Kerberos users logged in!
  valid users =  samba_user1
```

More security stuff:

The file /etc/samba/smb.secrets stores encrypted passwords (machine, keys). Use alternative approaches within /etc/samba/smb.conf first. If using Kerberos keytabs, this file can be emptied.

firewalld configuration: sudo firewall-cmd --permanent --add-service=samba
iptables configuration: sudo iptables -I INPUT -p tcp --dport 137-139,445 -j ACCEPT
List SELinux stuff with semanage fcontext -l | grep -i '(smb_|samba_)' semanage boolean -l | grep -i '(smb_|samba_)'

Extending SMB: The "Samba-VFS" framework

Is not a virtual file system as named. Leverage to enhance SMB server with modules, shared libraries (.so files- think ldd, not kernel libraries like .ko) and they can be declared in /etc/samba/smb.conf using the vfs objects parameter. Some modules can be chained, allowing multiple modules to work sequentially.

```
[global]
  vfs objects = full_audit  # Load the vfs_full_audit module-  detailed logs of file operations for enhanced security
  vfs_full_audit_log_dir = /var/log/samba/audit  # Specify log directory
  vfs_full_audit_log_file = full_audit.log    # Specify log filename
  vfs_full_audit_log_rotate = 5            # Rotate logs after 5 rotations
  vfs_full_audit_log_size = 10M              # Maximum log size (10 Megabytes)
```

```
[global]
  vfs objects = acl_tdb  #  Access control through storage of Access Control Lists (ACLs)
  vfs_acl_tdb_path = /etc/samba/acl.tdb     # Specify TDB database path.
```

vfs_recycle (recycle bin to recover deleted files); vfs_usershare (give users share definitions); vfs_fruit: (Apple File System (AFP) macOS client shares); vfs_fake_chroot (make a chroot environment for each connected user); vfs_deny_hosts (restrict access to shares based on IP or hostnames); vfs_cifs_xattr (enables storing extended attributes on Samba shares)