

# NFS Server Task: Professional Project Document

NAME : Ziad Mahmoud Ahmed Abdelgwad

## Objective:

Simulate an NFS server and client setup on a single Linux machine using localhost. Configure a shared directory, mount it locally, and test file sharing.

## Step 1: Install and Configure the NFS Server

The first step is to install the necessary NFS server packages and verify that the service is running correctly.

### 1.1. Install the NFS server package:

The following commands were used to update the package list and install the NFS kernel server on a Debian-based system (Kali Linux).

#### Bash

```
sudo apt-get update
```

```
sudo apt-get install nfs-kernel-server
```

```
(ziad@ziad)~$ sudo apt-get update
[sudo] password for ziad:
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [21.0 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [51.4 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [117 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [327 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [198 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [911 kB]
Get:8 http://kali.download/kali kali-rolling/non-free-firmware amd64 Packages [10.8 kB]
Get:9 http://kali.download/kali kali-rolling/non-free-firmware amd64 Contents (deb) [26.7 kB]
Fetched 74.0 MB in 15s (4,825 kB/s)
Reading package lists... Done

(ziad@ziad)~$ sudo apt-get install nfs-kernel-server
```

### 1.2. Check the service status:

After installation, the systemctl command was used to verify that the nfs-server service is active and running.

#### Bash

```
sudo systemctl status nfs-server
```

```
(ziad@ziad)~$ sudo systemctl status nfs-server
● nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; preset: disabled)
   Active: active (exited) since Sun 2025-08-03 03:11:10 EDT; 1min 11s ago
  Invocation: d7c0e1cc5b9f4d1c95bd94cee76204ea
     Docs: man:rpc.nfsd(8)
           man:exportfs(8)
   Process: 3153 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
   Process: 3154 ExecStart=/bin/sh -c /usr/sbin/nfsdctl autostart || /usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
  Main PID: 3154 (code=exited, status=0/SUCCESS)
    Mem peak: 1.8M
       CPU: 12ms

Aug 03 03:11:10 ziad systemd[1]: Starting nfs-server.service - NFS server and services ...
Aug 03 03:11:10 ziad sh[3156]: nfsdctl: lockd configuration failure
Aug 03 03:11:10 ziad systemd[1]: Finished nfs-server.service - NFS server and services.

(ziad@ziad)~$
```

## Step 2: Create and Export the Shared Directory

This step involves creating the shared directory, setting its permissions, and configuring the NFS server to export it to the local machine.

### 2.1. Create the shared directory:

A new directory `/srv/nfs_server` was created, and its ownership and permissions were adjusted to ensure the NFS service can access it.

**Bash**

```
sudo mkdir -p /srv/nfs_server
```

```
sudo chown nobody:nogroup /srv/nfs_server
```

```
sudo chmod 777 /srv/nfs_server
```

```
(ziad@ziad)-[~]
$ sudo mkdir -p /srv/nfs_server

(ziad@ziad)-[~]
$ sudo chown nobody:nogroup /srv/nfs_server

(ziad@ziad)-[~]
$ sudo chmod 777 /srv/nfs_server

(ziad@ziad)-[~]
```

### 2.2. Edit the `/etc/exports` file:

The nano text editor was used to open the `/etc/exports` file.

**Bash**

```
sudo nano /etc/exports
```

```
(ziad@ziad)-[~]
$ sudo nano /etc/exports
```

The following line was added to the file to export the directory to the loopback IP address (127.0.0.1) with read/write (rw) access and other options for stable sharing.

```
File Actions Edit View Help
GNU nano 8.3 /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
# to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
/srv/nfs_server 127.0.0.1(rw,sync,no_subtree_check)

^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute
^X Exit ^R Read File ^N Replace ^U Paste ^J Justify ^C Location
M-U Undo M-E Redo
```

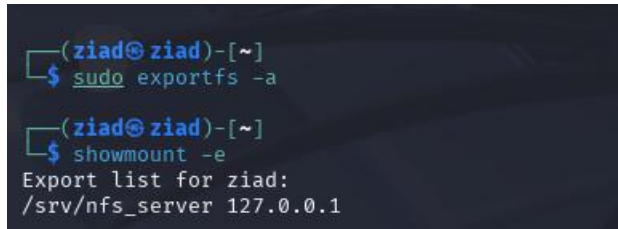
### 2.3. Export and verify the configuration:

The `exportfs -a` command was executed to apply the changes without a reboot, and `showmount -e` was used to confirm that the directory is correctly exported.

**Bash**

```
sudo exportfs -a
```

```
showmount -e
```



```
(ziad@ziad)-[~]  
$ sudo exportfs -a  
  
(ziad@ziad)-[~]  
$ showmount -e  
Export list for ziad:  
/srv/nfs_server 127.0.0.1
```

---

## Step 3: Simulate a Client and Mount the Directory

A separate directory was created to act as the client's mount point, and the NFS share was mounted to it.

### 3.1. Create the client mount point:

```
sudo mkdir -p /mnt/nfs_client
```

### 3.2. Mount the shared directory:

The NFS share was mounted using the server's loopback IP. The `mount | grep nfs` command was used to verify the successful mount.

```
sudo mount -t nfs 127.0.0.1:/srv/nfs_server /mnt/nfs_client
```

```
mount | grep nfs
```



```
ziad@ziad: ~  
File Actions Edit View Help  
  
(ziad@ziad)-[~]  
$ sudo mkdir -p /mnt/nfs_client  
  
(ziad@ziad)-[~]  
$ sudo mount -t nfs 127.0.0.1:/srv/nfs_server /mnt/nfs_client  
  
(ziad@ziad)-[~]  
$ mount | grep nfs  
nfsd on /proc/fs/nfsd type nfsd (rw,relatime)  
127.0.0.1:/srv/nfs_server on /mnt/nfs_client type nfs4 (rw,relatime,vers=4.2,rsize=1048576,wsiz=1048576,namlen=255,hard,proto=tcp,timeo=600,retra
```

## Step 4: Test File Sharing

To prove that the NFS setup is working, files were created on both the server and client paths.

### 4.1. Create a file on the server side:

The tee command was used with sudo to create a file with content in the server's shared directory.


```
echo "This is a file created on the server." | sudo tee /srv/nfs_server/server_file.txt
```

### 4.2. Verify the file on the client side:

The ls -l command on the client's mount point showed the new file. Its content was verified using cat.

```
ls -l /mnt/nfs_client
```

```
cat /mnt/nfs_client/server_file.txt
```



```
(ziad@ziad)-[~]
$ echo "This is a file created on the server." | sudo tee /srv/nfs_server/server_file.txt
This is a file created on the server.

(ziad@ziad)-[~]
$ ls -l /mnt/nfs_client
total 4
-rw-r--r-- 1 root root 38 Aug  3 03:26 server_file.txt

(ziad@ziad)-[~]
$ cat /srv/nfs_server/server_file.txt,
cat: /srv/nfs_server/server_file.txt,: No such file or directory

(ziad@ziad)-[~]
$ cat /srv/nfs_server/server_file.txt
This is a file created on the server.
```

---

## Step 5: Configure Persistent Mount

This section demonstrates how to make the NFS mount persistent across system reboots using `/etc/fstab`.

### 5.1. Edit the `/etc/fstab` file:

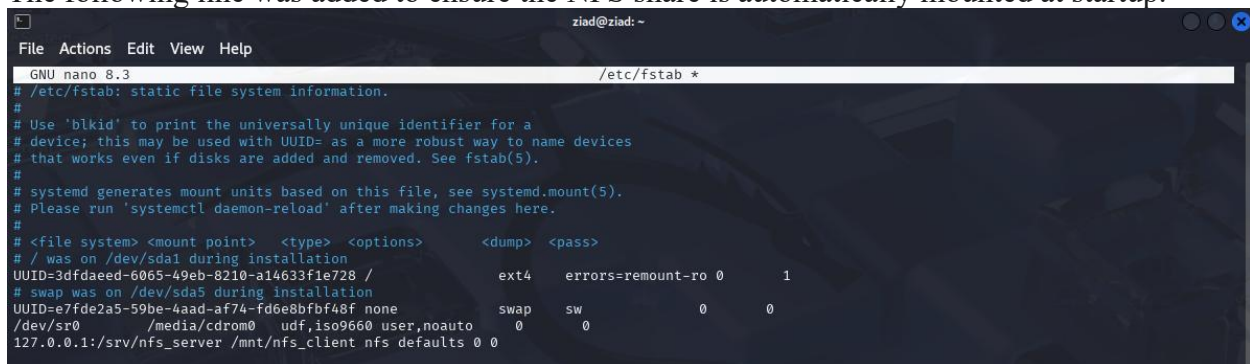
The file was opened with `nano`.

```
sudo nano /etc/fstab
```



```
(ziad@ziad)-[~]  
$ sudo nano /etc/fstab
```

The following line was added to ensure the NFS share is automatically mounted at startup.



```
File Actions Edit View Help  
GNU nano 8.3 /etc/fstab *  
# /etc/fstab: static file system information.  
#  
# Use 'blkid' to print the universally unique identifier for a  
# device; this may be used with UUID= as a more robust way to name devices  
# that works even if disks are added and removed. See fstab(5).  
#  
# systemd generates mount units based on this file, see systemd.mount(5).  
# Please run 'systemctl daemon-reload' after making changes here.  
#  
# <file system> <mount point> <type> <options> <dump> <pass>  
# / was on /dev/sda1 during installation  
UUID=3dfdaded-6065-49eb-8210-a14633f1e728 / ext4 errors=remount-ro 0 1  
# swap was on /dev/sda5 during installation  
UUID=e7fde2a5-59be-4aad-af74-fd6e8bfbf48f none swap sw 0 0  
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0  
127.0.0.1:/srv/nfs_server /mnt/nfs_client nfs defaults 0 0
```

### 5.2. Reload `systemd` and test the mount:

After adding the entry, `systemctl daemon-reload` was used to refresh the system's configuration cache, followed by `mount -a` to test the new `fstab` entry.

```
sudo umount /mnt/nfs_client
```

```
sudo systemctl daemon-reload
```

```
sudo mount -a
```



```
File Actions Edit View Help  
ziad@ziad: ~  
(ziad@ziad)-[~]  
$ sudo systemctl daemon-reload  
(ziad@ziad)-[~]  
$ sudo mount -a  
(ziad@ziad)-[~]  
$ mount | grep nfs  
nfsd on /proc/fs/nfsd type nfsd (rw,relatime)  
127.0.0.1:/srv/nfs_server on /mnt/nfs_client type nfs4 (rw,relatime,vers=4.2,rsize=1048576,wsiz=1048576,namlen=255,hard,proto=tcp,timeo=600,retra  
ns=2,sec=sys,clientaddr=127.0.0.1,local_lock=none,addr=127.0.0.1)  
(ziad@ziad)-[~]
```

## Step 6: Add a Firewall Rule

The final optional step involves adding a basic firewall rule to allow NFS traffic from the loopback IP address.

### 6.1. Install ufw:

As ufw was not found on the system, it was installed using apt-get.

```
sudo apt-get update
```

```
sudo apt-get install ufw
```

```
(ziad@ziad)-[~]
$ sudo apt-get update
sudo apt-get install ufw
Hit:1 http://http.kali.org/kali kali-rolling InRelease
Reading package lists ... Done
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
Suggested packages:
```

### 6.2. Configure and enable the firewall:

The rule to allow NFS from localhost was added, and then the firewall was enabled.

#### Bash

```
sudo ufw allow from 127.0.0.1 to any port nfs
```

```
sudo ufw enable
```

### 6.3. Check the firewall status:

The ufw status verbose command confirmed that the firewall is active and the rule for NFS traffic from 127.0.0.1 has been applied.

```
(ziad@ziad)-[~]
$ sudo ufw allow from 127.0.0.1 to any port nfs
Rules updated

(ziad@ziad)-[~]
$ sudo ufw allow from 127.0.0.1 to any port nfs
Skipping adding existing rule

(ziad@ziad)-[~]
$ sudo ufw enable
Firewall is active and enabled on system startup

(ziad@ziad)-[~]
$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To Action From
--
2049 ALLOW IN 127.0.0.1
```