Linux > Networking > NFS

Network File Sharing

SUSE Linux Enterprise Server SLES 12

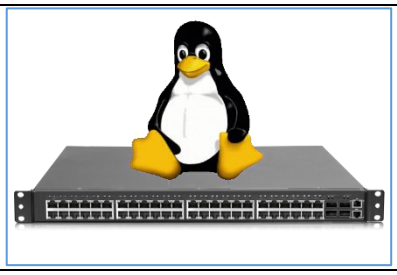


Table of Contents

1. Sharing File Systems with NFS

Distributing and sharing file systems over a network is a common task in corporate environments. The well-proven network file system (NFS) works with NIS, the yellow page's protocol. For a more secure protocol that works with LDAP and Kerberos, check NFSv4 (default). Combined with pNFS, you can eliminate performance bottlenecks.

NFS with NIS makes a network transparent to the user. With NFS, it is possible to distribute arbitrary file systems over the network. With an appropriate setup, users always find themselves in the same environment regardless of the terminal they currently use.

IMPORTANT: Need for DNS

In principle, all exports can be made using IP addresses only. To avoid time-outs, you need a working DNS system. DNS is necessary at least for logging purposes, because the mountd daemon does reverse lookups.

1.1. NFS Protocol Versions

The following are terms used in the YaST module.

- Exports

A directory exported by an NFS server, which clients can integrate it into their system.

- NFS Client

The NFS client is a system that uses NFS services from an NFS server over the Network File System protocol. The TCP/IP protocol is already integrated into the Linux kernel; there is no need to install any additional software.

- NFS Server

The NFS server provides NFS services to clients. A running server depends on the following daemons: nfsd (worker), idmapd (user and group name mappings to IDs and vice versa), statd (file locking), and mountd (mount requests).

- NFSv3

NFSv3 is the version 3 implementation, the old stateless NFS that supports client authentication.

- NFSv4

NFSv4 is the new version 4 implementation that supports secure user authentication via kerberos. NFSv4 requires one single port only and thus is better suited for environments behind a firewall than NFSv3.
The protocol is specified as http://tools.ietf.org/html/rfc3530.

Daemons for NFSv4

	client side	both sides	server side
user commands:	mount exportfs		
user daemons:		portmap idmapd	nfsd
kernel parts:		NFSv4 RPC XDR TCP Ipv4	

The following are the Daemons that should be running on a NFSv4 Server:

- **rpc.idmapd**
- **rpc.nfsd 8**

The following are the Daemons that should be running on a NFSv4 client:

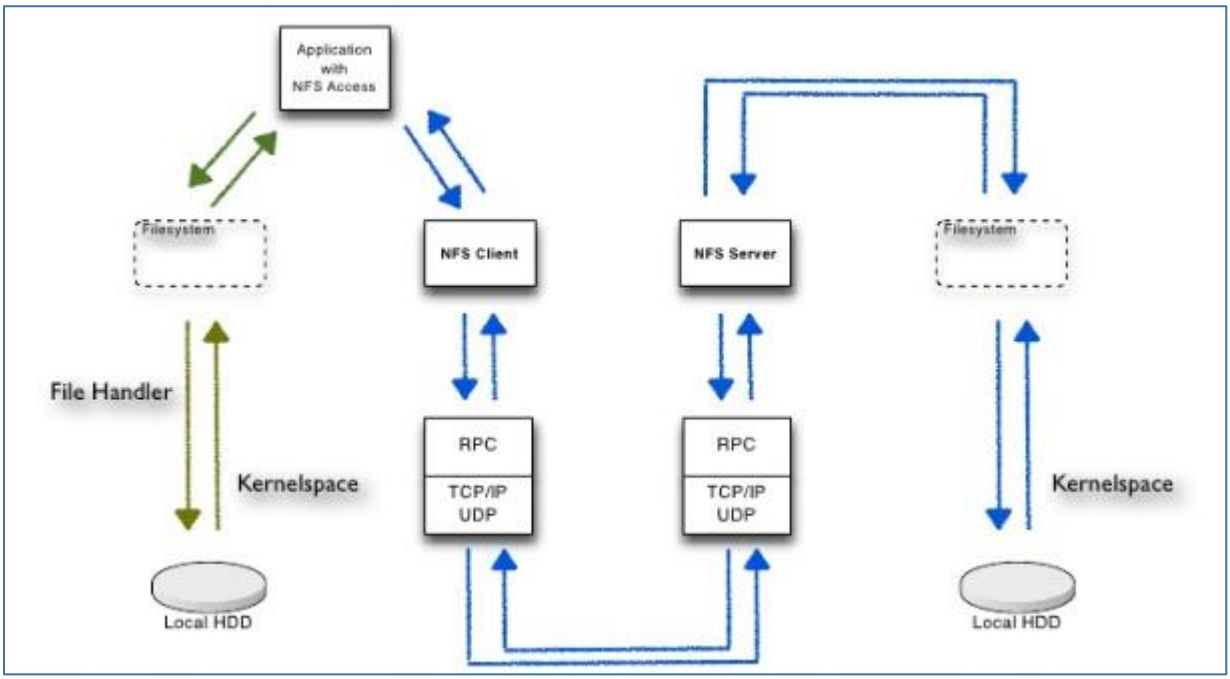
- **rpc.idmapd**

- pNFS

Parallel NFS, a protocol extension of NFSv4. Any pNFS clients can directly access the data on an NFS server.

1.2. NFS and RPC

NFS RFC Model



1.3. RPCINFO

rpcinfo makes an RPC call to an RPC server and reports what it finds.

In the first synopsis (summary), rpcinfo lists all the registered RPC services with rpcbind on host. If host is not specified, the localhost is the default. If -s is used, the information is displayed in a concise format.

In the second synopsis, rpcinfo lists all the RPC services registered with rpcbind, version 2. Also note that the format of the information is different in the first and the second synopsis. This is because the second synopsis is an older protocol used to collect the information displayed (version 2 of the rpcbind protocol).

The third synopsis makes an RPC call to procedure 0 of prognum and versnum on the specified host and reports whether a response was received. transport is the transport which has to be used for contacting the given service. The remote address of the service is obtained by making a call to the remote rpcbind.

The prognum argument is a number that represents an RPC program number. If a versnum is specified, rpcinfo attempts to call that version of the specified prognum. Otherwise, rpcinfo attempts to find all the registered version numbers for the specified prognum by calling version 0, which is presumed not to exist; if it does exist, rpcinfoattempts to obtain this information by calling an extremely high version number instead, and attempts to call each registered version.

The version number is required for -b and -d options.

```
linsrv3:~ # rpcinfo
   program version netid     address                service    owner
    100000    4    udp6      ::.0.111               portmapper superuser
    100000    3    udp6      ::.0.111               portmapper superuser
    100000    4    tcp6      ::.0.111               portmapper superuser
    100000    3    tcp6      ::.0.111               portmapper superuser
    100000    4    udp       0.0.0.0.0.111          portmapper superuser
    100000    3    udp       0.0.0.0.0.111          portmapper superuser
    100000    2    udp       0.0.0.0.0.111          portmapper superuser
    100000    4    tcp       0.0.0.0.0.111          portmapper superuser
    100000    3    tcp       0.0.0.0.0.111          portmapper superuser
    100000    2    tcp       0.0.0.0.0.111          portmapper superuser
    100000    4    local     /run/rpcbind.sock      portmapper superuser
    100000    3    local     /run/rpcbind.sock      portmapper superuser
```

rpcinfo -p localhost

2. Server Configuration

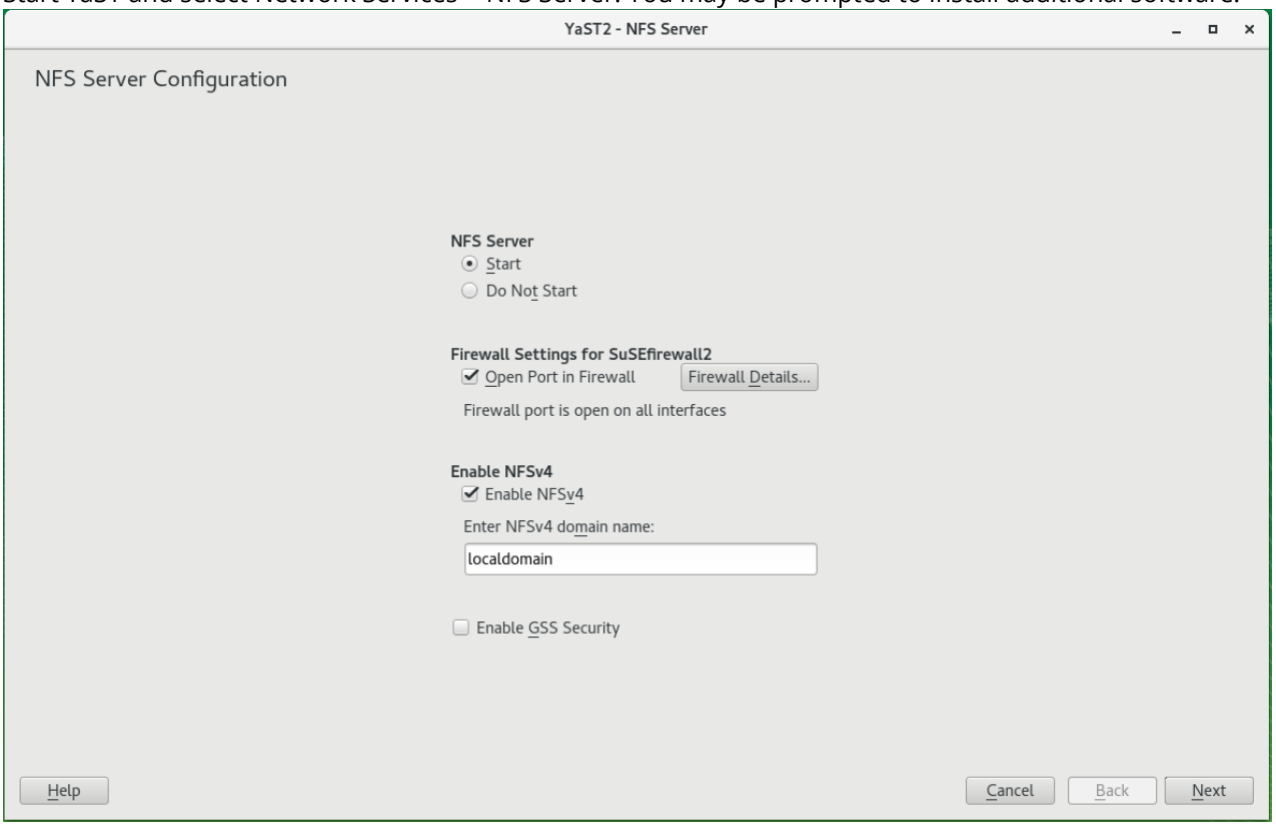
Configuring an NFS server can be done either through YaST or manually.
For authentication, NFS can also be combined with Kerberos.

Like NIS, NFS is a client/server system. However, a machine can be both—it can supply file systems over the network (export) and mount file systems from other hosts (import).

1.1. Exporting File Systems with YaST

With YaST, turn a host in your network into an NFS server—a server that exports directories and files to all hosts granted access to it or to all members of a group. Thus, the server can also provide applications without installing the applications locally on every host.

Start YaST and select Network Services > NFS Server. You may be prompted to install additional software.



1. Activate the Start radio button.

2. If a firewall is active on your system (SuSEFirewall2), check Open Ports in Firewall. YaST adapts its configuration for the NFS server by enabling the nfs service.

3. Check whether you want to Enable NFSv4. If you deactivate NFSv4, YaST will only support NFSv3. For information about enabling NFSv2, see NFSv2.

   1.1. If NFSv4 is selected, additionally enter the appropriate NFSv4 domain name.

   Make sure the name is the same as the one in the /etc/idmapd.conf file of any NFSv4 client that accesses this particular server. This parameter is for the idmapd daemon that is required for NFSv4 support (on both server and client). Leave it as localdomain (the default) if you do not have any special requirements.

4. Click Enable GSS Security if you need secure access to the server. A prerequisite for this is to have Kerberos installed on your domain and to have both the server and the clients kerberized. Click Next to proceed with the next configuration dialog.

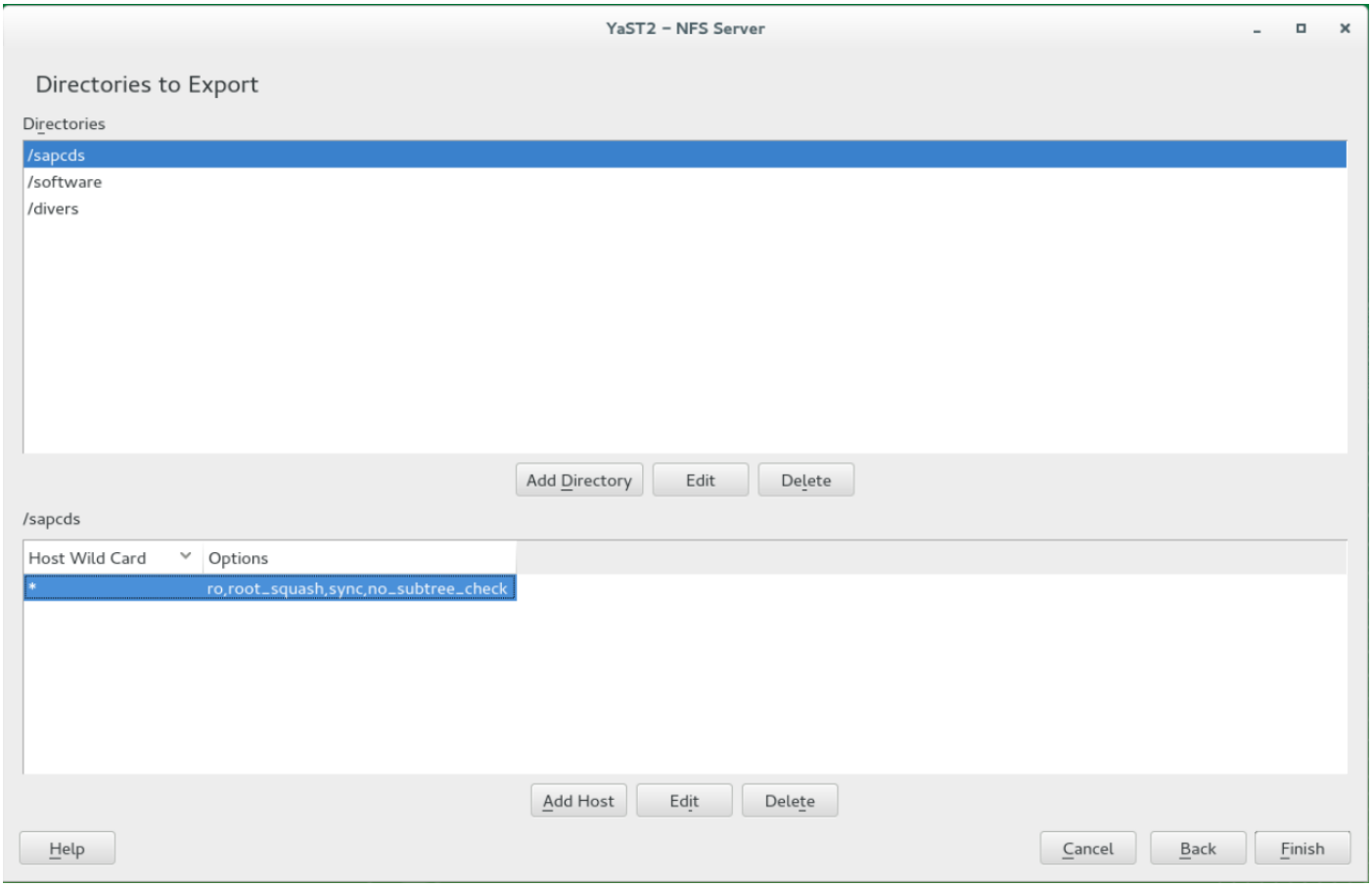
5. Click Add Directory in the upper half of the dialog to export your directory

6. If you have not configured the allowed hosts already, another dialog for entering the client information and options pops up automatically. Enter the host wild card (usually you can leave the default settings as they are).

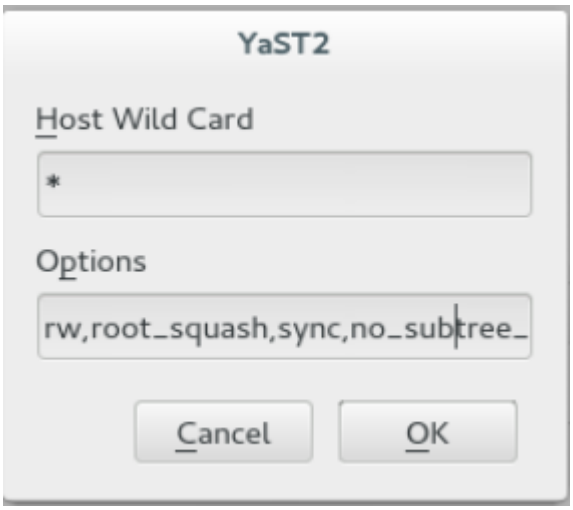
   There are four possible types of host wild cards that can be set for each host: a single host (name or IP address), netgroups, wild cards (such as * indicating all machines can access the server), and IP networks.

   For more information about these options, see the exports man page.

7. Click Finish to complete the configuration.



1.2. Configuring the Options



/sapcds is the directory to share "*" = Host Wild Card

ro => rw root_squash => no_root_squash

ro — Mounts of the exported file system are read-only. Remote hosts are not able to make changes to the data shared on the file system. To allow hosts to make changes to the file system, the read/write (rw) option must be specified.

wdelay — Causes the NFS server to delay writing to the disk if it suspects another write request is imminent. This can improve performance by reducing the number of times the disk must be accessed by separate write commands, reducing write overhead. The no_wdelay option turns off this feature, but is only available when using the sync option.

root_squash — Prevents root users connected remotely from having root privileges and assigns them the user ID for the user nfsnobody. This effectively "squashes" the power of the remote root user to the lowest local user, preventing unauthorized alteration of files on the remote server. Alternatively, the no_root_squash option turns off root squashing.
To squash every remote user, including root, use theall_squash option.
To specify the user and group IDs to use with remote users from a particular host, use the anonuid and anongid options, respectively.
In this case, a special user account can be created for remote NFS users to share and specify (anonuid=<uid-value>,anongid=<gid-value>), where <uid-value>is the user ID number and <gid-value> is the group ID number.

Each default for every exported file system must be explicitly overridden. For example, if the **`rw`** option is not specified, then the exported file system is shared as read-only. The following is a sample line from **`/etc/exports`** which overrides two default options:

```
/another/exported/directory 192.168.0.3(rw,sync)
```

In this example 192.168.0.3 can mount /another/exported/directory/ read/write and all transfers to disk are committed to the disk before the write request by the client is completed.

Additionally, other options are available where no default value is specified. These include the ability to disable sub-tree checking, allow access from insecure ports, and allow insecure file locks (necessary for certain early NFS client implementations). Refer to the exports man page for details on these lesser used options.

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.
For example, the following two lines do not mean the same thing:

```
/home bob.example.com(rw) /home bob.example.com (rw)
```

The first line allows only users from bob.example.com read/write access to the /home directory. The second line allows users from bob.example.com to mount the directory as read-only (the default), while the rest of the world can mount it read/write.

1.3. Exporting File Systems Manually

The configuration files for the NFS export service are /etc/exports and /etc/sysconfig/nfs. In addition to these files, /etc/idmapd.conf is needed for the NFSv4 server configuration. To start or restart the services, run the command systemctl restart nfsserver. This also starts the rpc.idmapd if NFSv4 is configured in /etc/sysconfig/nfs. The NFS server depends on a running RPC portmapper. Therefore, it also starts or restarts the portmapper service.

/etc/exports

The `/etc/exports` file contains a list of entries. Each entry indicates a directory that is shared and how it is shared. A typical entry in `/etc/exports` consists of:*`/shared/directory  host(option_list)`*
For example:`/export/data  192.168.1.2(rw,sync)`

```
linsrv3:~ # cat /etc/exports
/sapcds *(rw,no_root_squash,sync,no_subtree_check)
/software       *(rw,no_root_squash,sync,no_subtree_check)
/divers *(ro,root_squash,sync,no_subtree_check)
```

/etc/sysconfig/nfs

The /etc/sysconfig/nfs file contains a few parameters that determine NFSv4 server daemon behaviour. It is important to set the parameter NFS4_SUPPORT to yes(default). NFS4_SUPPORT determines whether the NFS server supports NFSv4 exports and clients.

/etc/idmapd.conf

Every user on a Linux machine has a name and an ID. idmapd does the name-to-ID mapping for NFSv4 requests to the server and replies to the client. It must be running on both server and client for NFSv4, because NFSv4 uses only names for its communication.

Make sure that there is a uniform way in which user names and IDs (uid) are assigned to users across machines that might probably be sharing file systems using NFS. This can be achieved by using NIS, LDAP, or any uniform domain authentication mechanism in your domain.

The parameter Domain must be set the same for both, client and server in the /etc/idmapd.conf file. If you are not sure, leave the domain as localdomain in the server and client files. A sample configuration file looks like the following:

```
linsrv3:~ # cat /etc/idmapd.conf
[General]

Verbosity = 0
Pipefs-Directory = /var/lib/nfs/rpc_pipefs
Domain = localdomain

[Mapping]

Nobody-User = nobody
Nobody-Group = nobody
```

3. Client Configuration

To configure your host as an NFS client, you do not need to install additional software. All needed packages are installed by default.

3.1. Importing File Systems with YaST

Authorized users can mount NFS directories from an NFS server into the local file tree using the YaST NFS client module.

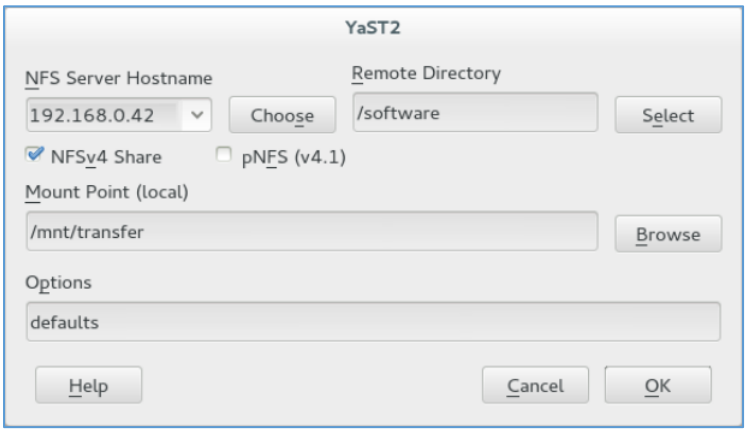
Importing NFS Directories

1. Start the YaST NFS client module.

2. Click *Add* in the *NFS Shares* tab. Enter the host name of the NFS server, the directory to import, and the mount point at which to mount this directory locally.
3. When using NFSv4, select *Enable NFSv4* in the *NFS Settings* tab. Additionally, the *NFSv4 Domain Name* must contain the same value as used by the NFSv4 server. The default domain is `localdomain`.
4. To use Kerberos authentication for NFS, GSS security must be enabled. Select *Enable GSS Security*.
5. Enable *Open Port in Firewall* in the *NFS Settings* tab if you use a Firewall and want to allow access to the service from remote computers. The firewall status is displayed next to the check box.
6. Click *OK* to save your changes.

The configuration is written to `/etc/fstab` and the specified file systems are mounted. When you start the YaST configuration client later, it also reads the existing configuration from this file.

HINT: NFS as a Root File System
On (diskless) systems where the root partition is mounted via network as an NFS share, you need to be careful when configuring the network device with which the NFS share is accessible.
When shutting down or rebooting the system, the default processing order is to turn off network connections, then unmount the root partition. With NFS root, this order causes problems as the root partition cannot be cleanly unmounted as the network connection to the NFS share is already not activated. To prevent the system from deactivating the relevant network device, open the network device configuration tab as described in Activating the Network Device, and choose On NFSroot in the Device Activation pane.



3.2. Importing File Systems Manually

The prerequisite for importing file systems manually from an NFS server is a running RPC port mapper. The nfs service takes care to start it properly; thus, start it by entering `systemctl start nfs` as root. Then remote file systems can be mounted in the file system like local partitions using `mount`:

```
mount linsrv3.terwal.local:/sapcds /mnt/transfer
```

To import user directories from the linsrv3.terwal.local machine, for example, use:

```
mount linsrv3.terwal.local:/home /home
```

3.3. Using the Automount Service

The autofs daemon can be used to mount remote file systems automatically. Add the following entry to the /etc/auto.master file:

```
/nfsmounts /etc/auto.nfs
```

Now the /nfsmounts directory acts as the root for all the NFS mounts on the client if the auto.nfs file is filled appropriately. The name auto.nfs is chosen for the sake of convenience—you can choose any name. In auto.nfs add entries for all the NFS mounts as follows:

```
localdata -fstype=nfs server1:/data
nfs4mount -fstype=nfs4 server2:/
```

Activate the settings with systemctl start autofs as root. In this example, /nfsmounts/localdata, the /data directory of server1, is mounted with NFS and/nfsmounts/nfs4mount from server2 is mounted with NFSv4.

If the /etc/auto.master file is edited while the service autofs is running, the automounter must be restarted for the changes to take effect with systemctl restart autofs.

Manually Editing /etc/fstab

A typical NFSv3 mount entry in /etc/fstab looks like this:

```
nfs.example.com:/data  /local/path nfs rw,noauto 0 0
```

For NFSv4 mounts, use nfs4 instead of nfs in the third column:

```
nfs.example.com:/data  /local/pathv4 nfs4 rw,noauto 0 0
```

The noauto option prevents the file system from being mounted automatically at start-up. If you want to mount the respective file system manually, it is possible to shorten the mount command specifying the mount point only:
mount /local/path

NOTE: Mounting at Start-Up
If you do not enter the noauto option, the init scripts of the system will handle the mount of those file systems at start-up.

4. Operations

Linsrv3:~ # systemctl stop nfsserver

```
2016-12-20T19:19:14.493542+01:00 linsrv3 systemd[1]: Stopping NFS server and services...
2016-12-20T19:19:14.494759+01:00 linsrv3 systemd[1]: Stopping Alias for NFS server...
2016-12-20T19:19:14.495604+01:00 linsrv3 systemd[1]: Stopped Alias for NFS server.
2016-12-20T19:19:14.506559+01:00 linsrv3 systemd[1]: Stopped NFS server and services.
2016-12-20T19:19:14.507182+01:00 linsrv3 systemd[1]: Stopping NFSv4 ID-name mapping service...
2016-12-20T19:19:14.507476+01:00 linsrv3 systemd[1]: Stopping NFS Mount Daemon...
2016-12-20T19:19:14.508089+01:00 linsrv3 kernel: [ 8084.160307] nfsd: last server has exited, flushing export cache
2016-12-20T19:19:14.507916+01:00 linsrv3 rpc.mountd[13588]: Caught signal 15, un-registering and exiting.
2016-12-20T19:19:14.508516+01:00 linsrv3 systemd[1]: Stopped NFSv4 ID-name mapping service.
2016-12-20T19:19:14.508717+01:00 linsrv3 systemd[1]: Stopped NFS Mount Daemon.
2016-12-20T19:19:14.511006+01:00 linsrv3 systemd[1]: Stopping Preprocess NFS configuration...
2016-12-20T19:19:14.511200+01:00 linsrv3 systemd[1]: Stopped Preprocess NFS configuration.
```

Linsrv3:~ # systemctl start nfsserver

```
2016-12-20T19:25:46.805549+01:00 linsrv3 systemd[1]: Started Kernel Module supporting RPCSEC_GSS.
2016-12-20T19:25:46.805784+01:00 linsrv3 systemd[1]: Starting Preprocess NFS configuration...
2016-12-20T19:25:46.807164+01:00 linsrv3 systemd[1]: Starting Alias for NFS server...
2016-12-20T19:25:46.810398+01:00 linsrv3 systemd[1]: Started Alias for NFS server.
2016-12-20T19:25:46.817684+01:00 linsrv3 systemd[1]: Started Preprocess NFS configuration.
2016-12-20T19:25:46.817969+01:00 linsrv3 systemd[1]: Starting NFS Mount Daemon...
2016-12-20T19:25:46.818338+01:00 linsrv3 systemd[1]: Starting NFSv4 ID-name mapping service...
2016-12-20T19:25:46.818829+01:00 linsrv3 systemd[1]: Started RPC security service for NFS client and server.
2016-12-20T19:25:46.819012+01:00 linsrv3 systemd[1]: Started RPC security service for NFS server.
2016-12-20T19:25:46.820984+01:00 linsrv3 systemd[1]: Started NFSv4 ID-name mapping service.
2016-12-20T19:25:46.823512+01:00 linsrv3 rpc.mountd[13830]: Version 1.3.0 starting
2016-12-20T19:25:46.823855+01:00 linsrv3 systemd[1]: Started NFS Mount Daemon.
2016-12-20T19:25:46.824128+01:00 linsrv3 systemd[1]: Starting NFS server and services...
2016-12-20T19:25:46.837528+01:00 linsrv3 systemd[1]: Started NFS server and services.
2016-12-20T19:25:46.837728+01:00 linsrv3 systemd[1]: Starting Notify NFS peers of a restart...
2016-12-20T19:25:46.839681+01:00 linsrv3 sm-notify[13845]: Version 1.3.0 starting
2016-12-20T19:25:46.839860+01:00 linsrv3 sm-notify[13845]: Already notifying clients; Exiting!
2016-12-20T19:25:46.840028+01:00 linsrv3 kernel: [ 8476.420783] NFSD: starting 90-second grace period (net ffffffff81cf5e40)
2016-12-20T19:25:46.840741+01:00 linsrv3 systemd[1]: Started Notify NFS peers of a restart.
```