Oracle Linux 9

Managing the Network File System

G26606-03
September 2025

Oracle Linux 9 Managing the Network File System,

G26606-03

Contents

Preface

This chapter includes information about managing the Network File System (NFS) in Oracle Linux 9, including tasks for configuring, administering, and using NFS.

1 About NFS

NFS (Network File System) is a distributed file system that lets a client system access files over a network as though the files were on local storage.

An NFS server can share directory hierarchies in its local file systems with remote client systems over an IP-based network. After an NFS server exports a directory, NFS clients with the appropriate permissions can mount this directory. To the client systems, the directory appears as if it's a local directory.

The benefits of using NFS include centralized storage provisioning, improved data consistency, and reliability.

Enabled Versions of NFS

The following versions of NFS are enabled in Oracle Linux:

- NFS version 3 (NFSv3), specified in RFC 1813.
- NFS version 4 (NFSv4), specified in RFC 7530.
- NFS version 4 minor version 1 (NFSv4.1), specified in RFC 8881.
- NFS version 4 minor version 2 (NFSv4.2), specified in RFC 7862 .

NFSv3 provides safe, asynchronous writes, and efficient error handling. NFSv3 also uses 64-bit file sizes and offsets, which enable clients to access more than 2 GB of file data.

NFSv3 relies on Remote Procedure Call (RPC) services, which are controlled by the `rpcbind` service. The `rpcbind` service responds to requests for an RPC service and then sets up connections for the requested service. Separate services are used to handle locking and mounting protocols. Configuring a firewall to cope with the various ports that are used by all these services can be complex and error-prone.

NFSv4:

- Works across firewalls and the Internet.
- Doesn't require the `rpcbind` service.
- Uses Access Control Lists (ACLs).
- Uses stateful operations.

NFSv4 requires the Transmission Control Protocol (TCP) running over an IP network. Instead of using `rpcbind`, the NFS server listens on TCP port 2049 for service requests. The mounting and locking protocols are also integrated into the NFSv4 protocol, which means that separate services are also not required for these protocols. These refinements make firewall configuration for NFSv4 no more difficult than for a service such as HTTP.

NFS clients mount by using NFSv4.2 (the default version), but fall back to NFSv4.1 when the server doesn't work with NFSv4.2. The mount later falls back to NFSv4.0 and then to NFSv3.

About NFS Services

The NFS versions used by Oracle Linux 9 rely on Remote Procedure Calls (RPC) between clients and servers. To share or mount NFS file systems, the following required services work together, depending on which version of NFS is implemented. Note that all these services are started automatically:

nfsd
Server kernel module that services requests for shared NFS file systems.

rpcbind
Acts as a central "directory" service for NFS (and other RPC) services. Because some NFS services use dynamic ports, clients contact rpcbind on the server to find out which ports to use.

rpc.mountd
Process that's used by an NFS server to process mount requests from NFSv3 clients. The service checks that the requested NFS share is exported by the NFS server.

rpc.nfsd
Process that lets explicit NFS versions and protocols the server advertises to be defined.

lockd
Kernel thread that runs on both clients and servers. The `lockd` process implements the Network Lock Manager (NLM) protocol, which lets NFSv3 clients lock files on the server. The daemon is started automatically whenever the NFS server is run and whenever an NFS file system is mounted.

rpc-statd
Process that implements the Network Status Monitor (NSM) RPC protocol, which notifies NFS clients when an NFS server is restarted without first being brought down. The `rpc-statd` service is automatically started by the `nfs-server` service. This service doesn't require configuration by the user and isn't used with NFSv4.

rpc-idmapd
Process that provides NFSv4 client and server upcalls, which map between on-the-wire NFSv4 names (strings in the form of `user@domain`) and local UIDs and GIDs. The ID mapping service isn't enabled by default and must be explicitly started when using NFSv4. For the `idmapd` process to operate correctly, you must configure the `/etc/idmapd.conf` file with the appropriate domain and settings for the network. Note that only NFSv4 uses the `rpc-idmapd` service, earlier NFS versions don't require it, because they use numeric IDs directly.

> **Note**
>
> The mounting and locking protocols are incorporated into the NFSv4 protocol. Also, the server listens on TCP port 2049. For this reason, NFSv4 doesn't need to interact with the `rpcbind`, `lockd`, and `rpc-statd` services. However, the `nfs-mountd` service is still required to set up exports on the NFS server; but, the service isn't involved in any over the wire operations.
>
> The `rpc-idmapd` service only handles upcalls from the kernel and isn't itself directly involved in any over the wire operations. The service, however, might make naming service calls, which do result in over the wire lookups.

2

Configuring an NFS Server

You configure an NFS server in Oracle Linux 9 by first editing the /etc/exports file to grant directory access to NFS clients, and then making those shared directories available using the exportfs command.

Editing the /etc/exports File

The following steps describe how to configure shared directories using the /etc/exports file.

Configure the directories that an NFS server exports, including which clients can access those directories and what permissions they have, by editing the /etc/exports file.

> **Note**
>
> You can also configure exports in files that you create under the /etc/exports.d directory. For example, /etc/exports.d/myexports.

1. Install nfs-utils.

   If it's not already installed, install the nfs-utils package.

   ```
   sudo dnf install nfs-utils
   ```

2. Configure the /etc/exports file.

   Edit the /etc/exports file to define the directories that the server makes available for clients to mount, for example:

   ```
   /var/folder 192.0.2.102(rw,async)
   /usr/local/apps *(all_squash,anonuid=501,anongid=501,ro)
   /var/projects/proj1 192.168.1.0/24(ro) mgmtpc(rw)
   ```

   Each entry includes the local path to the exported directory, followed by a list of clients that can mount the directory and then client-specific export options (in parentheses). There can't be any spaces between the client specifier and the parenthesized list of options that apply to that client.

   The following information explains the example export file entries in greater detail:

   - Only the client system with the IP address 192.0.2.102 can mount the /var/folder directory with read and write permissions. All writes to the disk are asynchronous. This means that the server doesn't wait for write requests to be written to disk before responding to further requests from the client.
   - As indicated by the wildcard (*), all clients can mount the /usr/local/apps directory as read-only. All connecting users, including root users, are mapped to the local, unprivileged user with UID 501 and GID 501.

- All clients on the 192.168.1.0/24 subnet can mount the /var/projects/proj1 directory as read-only. However, the client system named mgmtpc can mount the directory with read/write permissions.

For more information on the format of the etc/exports file, see the exports(5) manual page.

3. Configure the /etc/idmapd.conf file for NFSv4 clients.

   If the server serves NFSv4 clients, edit the /etc/idmapd.conf file's definition for the Domain parameter by specifying the server's domain name.

   ```
   Domain = mydom.com
   ```

   This setting prevents the owner and group from being incorrectly listed as the anonymous user or group (nobody or nogroup) on NFS clients when the all_squash mount option isn't specified.

4. Configure the firewall to enable access only for NFSv4 clients.

   To enable access through the firewall for NFSv4 clients only, use the following commands:

   ```
   sudo firewall-cmd --permanent --zone=zone --add-service=nfs
   ```

   This configuration assumes that rpc.nfsd listens for client requests on the default TCP port 2049.

5. Configure the firewall to enable access for NFSv3 and NFSv4 clients.

   To enable access through the firewall for NFSv3 and NFSv4 clients, do the following:

   a. Edit the /etc/nfs.conf file to specify the port settings for handling network mount requests (mountd section) and status monitoring (statd section). Also, set the TCP port on which the network lock manager listens in the lockd section. For example:

      ```
      # Ports that various services should listen on.

      [mountd]
      port = 892

      [statd]
      port = 662

      [lockd]
      port = 32803
      ```

      If any of these ports are already in use, NFS fails to start. Use the lsof -i command to find an unused port and then change the setting in the /etc/nfs.conf file as appropriate.

      To confirm on which ports RPC services are listening, use the rpcinfo -p command.

   b. Restart the firewall service and configure the firewall to let NFSv3 connections through:

      ```
      sudo firewall-cmd --permanent --zone=zone --add-port=2049/tcp --add-port=111/tcp --add-port=32803/tcp --add-port=892/tcp --add-port=662/tcp
      ```

c. Reboot the server.

```
sudo systemctl reboot
```

6. Start the `nfs-server` service.

   Start the `nfs-server` service and configure the service to start automatically when the system boots:

   ```
   sudo systemctl enable --now nfs-server
   ```

7. Verify which versions of NFS the server works with.

   Run the following command to check that the server provides the NFS versions that you have configured:

   ```
   sudo cat /proc/fs/nfsd/versions
   ```

   For example, the following output shows that the server provides NFSv3, NFSv4, NFSv4.1, and NFSv4.2:

   ```
   +3 +4 +4.1 +4.2
   ```

8. List the exported directories.

   Display a list of the exported directories.

   ```
   sudo showmount -e
   ```

   ```
   Export list for host01.mydom.com
   /var/folder 192.0.2.102
   /usr/local/apps *
   /var/projects/proj1 192.168.1.0/24 mgmtpc
   ```

   The `exportfs` command on the server displays the same information as the `showmount -e` command.

   ```
   sudo /usr/sbin/exportfs -v
   ```

   The `showmount -a` command displays all the current clients and all the exported directories that the clients have mounted.

   Note

   To enable use of the `showmount` command from NFSv4 clients, specify a port number to the `MOUNTD_PORT` parameter in `/etc/nfs.conf`. Then, create a firewall rule to enable access to this TCP port.

The exportfs command

Describes the `exportfs` command.

The exportfs command lets an administrator export or unexport directories selectively, without needing to restart the NFS service. When provided with the appropriate options, the exportfs command writes the exported directories to the /var/lib/nfs/etab file.

Changes to the list of exported directories are effective immediately because the nfs-mountd service refers to the etab file for a specific directory's access privileges.

Using the exportfs Command

If used without any options, the exportfs command displays a list of exported directories. Providing options to the exportfs command let you be selective about what gets exported.

The exportfs command options include the following:

-r
Refreshes the list of exported directories in the /var/lib/nfs/etab file by incorporating any changes that were made to the list in the /etc/exports file.

-a
Exports all the directories that are specified in the /etc/exports file. This option can be combined with other options, to specify what action is performed on the directories.

-u
Unexports one or more shared directories.

> **Note**
>
> The exportfs -ua command suspends NFS file sharing, but keeps all NFS services running. To reenable NFS sharing, use the exportfs -r command.

-v
Specifies verbose logging, which displays detailed information about the file systems that are being exported or unexported.

Example 2-1   Export all directories in the /etc/exports file

To export every directory share defined in the /etc/exports file:

```
exportfs -a
```

Example 2-2   Export a single directory from the /etc/exports file

To export only the /var/projects/proj1 directory from the /etc/exports file:

```
exportfs /var/projects/proj1
```

Example 2-3   Unexport a directory defined in the /etc/exports file

To unexport the /var/projects/proj1 directory from the /etc/exports file:

```
exportfs -u /var/projects/proj1
```

Example 2-4 Show detailed information about all exported directories

To show verbose information about all the directories being exported from the `/etc/exports` file:

```
exportfs -v
```

For more information on the `exportfs` command, see the `exportfs(8)`, `exports(5)`, and `showmount(8)` manual pages.

Configuring an NFS Server with TLS

This task shows how to set up the NFS server to use TLS encryption to secure data in transit and enable secure connections from trusted clients.

Ensure that the following are true:

- The system is running Oracle Linux 9 or later.
- You have configured the Oracle Linux system as an NFSv4 server.
- You can use an existing CA certificate or generate a self-signed certificate.
  - In production environments, obtain a TLS certificate and private key pair from the Certificate Authority (CA).
  - For testing and development only, you can use a self-signed certificate. First, follow the instructions in Generating a Self-Signed Certificate for TLS Authentication and then begin with the step to configure the NFS server for TLS by editing `/etc/tlshd.conf` that follows.
- You have installed the `ktls-utils` package.

1. Create a server private key and certificate signing request (CSR).

   Run the following command, replacing the Common Name (CN), DNS, and IP address with the server's actual host information.

   ```
   openssl req -new -newkey rsa:4096 -noenc \
      -keyout /etc/pki/tls/private/server.example.com.key \
      -out /etc/pki/tls/private/server.example.com.csr \
      -subj "/C=US/ST=State/L=City/O=Organization/CN=hostname" \
      -addext "subjectAltName=DNS:hostname,IP:host-ip-address"
   ```

2. Obtain a server certificate.
   - Send the generated CSR to the CA and request a signed certificate.
   - Store the returned CA certificate (`ca.crt`) and server certificate (`server.example.com.crt`) on the server.

3. Import the CA certificate into the system trust store.

   Move the certificate into the required location and update the trust store as follows:

   ```
   sudo cp ca.crt /etc/pki/ca-trust/source/anchors/
   sudo update-ca-trust
   ```

4. Install the server certificate.

Install the server certificate by moving it to the appropriate location in the file system:

```
sudo mv server.example.com.crt /etc/pki/tls/certs/
```

5. Restore SELinux contents.

   When you move or copy files such as certificates for NFS with TLS into security-sensitive directories, their SELinux labels might not match what's required for those locations. Run `restorecon` to ensure that the certificate files have the appropriate SELinux labels so that SELinux lets services access them.

   ```
   sudo restorecon -Rv /etc/pki/tls/certs/
   ```

6. Configure the NFS server for TLS.

   Edit `/etc/tlshd.conf` and add the following to the `[authenticate.server]` section:

   ```
   x509.certificate = /etc/pki/tls/certs/server.example.com.crt
   x509.private_key = /etc/pki/tls/private/server.example.com.key
   ```

   > **Note**
   >
   > Leave the `x509.truststore` parameter unset. The server doesn't need to verify client certificates unless mutual TLS authentication is being used.

7. Enable and start the TLS daemon.

   Run the following command to enable `tlshd` immediately and whenever the system reboots:

   ```
   sudo systemctl enable --now tlshd.service
   ```

The NFS server is now configured to work with TLS connections.

Generating a Self-Signed Certificate for TLS Authentication

For testing and development purposes only, you can use a self-signed certificate to configure NFS with TLS authentication.

This task shows how to generate a self-signed certificate.
Complete these steps on the NFS server.

1. Create the certificate and key.

   If the fully qualified domain name (FQDN) of the NFS server is 64 characters or shorter, run the following command:

   ```
   openssl req -noenc -x509 -newkey rsa:4096 -days 365 \
   -keyout nfsd.key -out nfsd.crt
   ```

   The command prompts you to enter values for several fields. Enter the FQDN as the Common Name (CN). You can leave all the other fields blank, or accept the defaults.

If the FQDN of the NFS server is longer than 64 characters, specify the FQDN as a Subject Alternative Name (SAN) at the command line, using the following syntax:

```
openssl req -noenc -x509 -copy_extensions copy \
-addext "subjectAltName = DNS:<FQDN of server>" \
-newkey rsa:4096 -days 365 -keyout nfsd.key -out nfsd.crt
```

Then, enter a shorter name (such as the plain, unqualified hostname) as the certificate CN when prompted.

> **Important**
>
> You can't use wildcards in the Common Name (CN) field.

2. Verify that the certificate is generated successfully.

   Run the following command to inspect the certificate and check that the output includes the correct CN, and SAN if specified:

   ```
   openssl x509 -in nfsd.crt -text -noout
   ```

3. Secure the generated certificate and key.

   Run the following command to change the ownership of the certificate and key to `root`:

   ```
   chown root:root nfsd.key nfsd.crt
   ```

3

Configuring NFS Clients

This section describes how to configure clients to access NFS shared directories over the network as if they were local directories.

Mounting an NFS Share

Describes how to mount an NFS share on a client.

For more information on mounting NFS shares, see the `mount(8)`, `nfs(5)`, and `showmount(8)` manual pages.

1. Install `nfs-utils`.

   If it's not already installed, install the `nfs-utils` package.

   ```
   sudo dnf install nfs-utils
   ```

2. List the NFS server's exported directories.

   Display a list of the directories that the NFS server exports. For example:

   ```
   sudo showmount -e host01.mydom.com
   ```

   The output of the previous command would be similar to the following:

   ```
   Export list for host01.mydom.com
   /var/folder 192.0.2.102
   /usr/local/apps *
   /var/projects/proj1 192.168.1.0/24 mgmtpc
   ```

   > **Note**
   >
   > Some servers don't accept querying the list of exports.

3. Mount an exported directory.

   Mount an exported NFS directory on an available mount point. For example:

   ```
   sudo mount -t nfs -r -o nosuid host01.mydoc.com:/usr/local/apps /apps
   ```

   This example mounts the `/usr/local/apps` directory that's exported by `host01.mydoc.com` with read-only permissions on `/apps`. The `nosuid` option prevents remote users from gaining greater privileges by running a `setuid` program.

> **Tip**
>
> Typically, the `-t` (or `--type`) `nfs` option can be omitted and the `mount` command guesses the file type.

4. (Optional) Mount the NFS share when the system boots.

   To configure the system to mount an NFS share at boot time, add an entry for the share to the `/etc/fstab` file, as shown in the following example:

   ```
   host01.mydoc.com:/usr/local/apps      /apps      nfs      ro,nosuid  0 0
   ```

Configuring an NFS Client with TLS

This task shows how to set up the NFS client to connect securely to a TLS enabled NFS server, ensuring data transmitted between client and server is encrypted.

Ensure that the following are true:

- The system is running Oracle Linux 9 or later.
- You have the Certificate Authority (CA) certificate from the NFS server, or have generated a self-signed certificate on the NFS server as described in Generating a Self-Signed Certificate for TLS Authentication.
- The `ktls-utils` package is installed.

1. (Optional) Remove obsolete certificates.

   If you have been using self-signed certificates for testing, it's a good idea to remove any previous, obsolete anchors first. For example, to list the existing anchors and then remove an unwanted anchor:

   ```
   trust list

   pkcs11:id=%43%0E%35%20%3B%78%60%39%D0%C7%F8%53%1A%B6%73%83%12%90%AC%5D;type
   =cert
   type: certificate
   label: Test CA
   trust: anchor
   category: authority
   ...
   sudo trust anchor --remove
   pkcs11:id=%43%0E%35%20%3B%78%60%39%D0%C7%F8%53%1A%B6%73%83%12%90%AC%5D;type
   =cert
   ```

2. Import the certificate into the system trust store.

   Run the following command to add the certificate as a new anchor in the system trust policy store:

   ```
   sudo trust anchor cert.pem
   ```

3. Enable and start the TLS daemon.

Run the following command to enable `tlshd` immediately and whenever the system reboots:

```
sudo systemctl enable --now tlshd.service
```

4. Mount the NFS share using TLS encryption.

   Run the following command, replacing `nfs-server` and `path/to/share` with the NFS server's host name and the exported directory:

   ```
   sudo mount -o xprtsec=tls server-hostname:/path/to/share /mnt/
   ```

5. Verify the connection.

   Run the following command. Look for a message in the output that says the server handshake was successful:

   ```
   sudo journalctl -u tlshd
   ```

Configuring an NFS Client with Mutual TLS Authentication

This task shows how to configure both the NFS client and server to authenticate each other using certificates during TLS encrypted connections. This ensures that only trusted systems can access NFS shares, providing extra security for sensitive or regulated environments.

Ensure that you have the following:

- An NFSv4 server and client, both with TLS enabled.
- The `ktls-utils` package installed on both systems.
- You have the Certificate Authority (CA) certificate from the NFS server, or have generated a self-signed certificate on the NFS server as described in Generating a Self-Signed Certificate for TLS Authentication. If you're using a self-signed certificate begin with the step to import the certificate into the system trust store that follows.

1. Create a client private key and CSR.

   Run the following command, substituting the correct values for Common Name (CN), DNS, and IP address:

   ```
   openssl req -new -newkey rsa:4096 -noenc \
      -keyout /etc/pki/tls/private/client.example.com.key \
      -out /etc/pki/tls/private/client.example.com.csr \
      -subj "/C=US/ST=State/L=City/O=Organization/CN=hostname" \
      -addext "subjectAltName=DNS:hostname,IP:host-ip-address"
   ```

2. Obtain a client certificate.

   - Send the generated CSR to the CA and request a signed certificate.
   - Store the returned CA certificate and client certificate (`client.example.com.crt`) on the client.

3. (Optional) Remove obsolete certificates.

If you have been using self-signed certificates for testing, it's a good idea to remove any previous, obsolete anchors first. For example, to list the existing anchors and then remove an unwanted anchor:

```
trust list

pkcs11:id=%43%0E%35%20%3B%78%60%39%D0%C7%F8%53%1A%B6%73%83%12%90%AC%5D;type
=cert
type: certificate
label: Test CA
trust: anchor
category: authority
...
sudo trust anchor --remove
pkcs11:id=%43%0E%35%20%3B%78%60%39%D0%C7%F8%53%1A%B6%73%83%12%90%AC%5D;type
=cert
```

4. Import the certificate into the system trust store.

   Run the following command to add the certificate as a new anchor in the system trust policy store:

   ```
   sudo trust anchor cert.pem
   ```

5. Configure the client for mutual TLS.

   Edit `/etc/tlshd.conf` and add the following under the `[authenticate.client]` section:

   ```
   x509.certificate = /etc/pki/tls/certs/client.example.com.crt
   x509.private_key = /etc/pki/tls/private/client.example.com.key
   ```

6. Enable and start the TLS daemon.

   Run the following command to enable `tlshd` immediately and whenever the system reboots:

   ```
   sudo systemctl enable --now tlshd.service
   ```

7. Mount the NFS share with mutual TLS.

   Run the following command, replacing `nfs-server` and `path/to/share` with the NFS server's host name and exported directory:

   ```
   sudo mount -o xprtsec=mtls nfs-server:/path/to/share /mnt/
   ```

8. Verify the connection.

   Run the following command. Look for a message in the output that says the server handshake was successful:

   ```
   sudo journalctl -u tlshd
   ```