NFS – Network File System

Linux uses paths and filenames to designate pretty much everything. You must know **what** a command needs, not just that it needs “some target with some slashes in it”

* /home/student is a directory
* /home/student/.bashrc is a file
* /dev/tty1 is a terminal
* //cstfile1 is a machine
* /dev/sda is a disk (physical device)
* /dev/sda1 is a partition
* /dev/team may be a volume group
* /dev/team/assets may be a logical volume
* /dev/null is nothing (black hole)

When you’re making notes on a command, specify not that it takes some kind of path, but **what** that path refers to.

Also, we want to be aware of the differences between **relative** paths and **absolute** paths. Relative paths are information pertaining to your current working directory. Absolute paths are from root (/)

Up to now, all the filesystem information we have been doing have been on local drives. Now we’re going to look at using file servers.

Linux uses a system called NFS, short for **Network File System**. This is a client-server system; a file server acts as the server, and the computer that wants to access the files acts as a client.

NFS allows a system to share directories and files with others over a network. By using NFS, users and programs can access files on remote systems almost as if they were local files.

Some of the most notable benefits that NFS can provide are:

* Local workstations use less disk space because commonly used data can be stored on a single machine and still remain accessible to others over the network
* There is no need for users to have separate home directories on every network machine. Home directories could be setup on the NFS server and made available through the network.
* Storage devices such as floppy disk, CDROM/DVD drives, USB Thumb drives can be used by other machines on the network. This may reduce the number of removable media drives throughout the network.

To begin, let’s work on a demo. In VMWare Workstation Pro, I’ll be creating two clones of my Ubuntu VM (1 will act as our NFS server, 1 will act as a client). We’ll reduce the CPU to 2 cores and RAM to 4GB.

We’ll also add a second network adapter to each clone, making sure to select “Host-only” instead of NAT.

|  |  |
| --- | --- |
|  |  |

Let’s change our hostnames so that they are A) distinct between the two clones and B) easier to connect to. We’ll use the naming convention <YourName>-NFS for our NFS server and <YourName>-NFSClient for our NFS Client.

To change our hostname, we’ll need to edit our /etc/hostname file.

sudo vi /etc/hostname

We’ll need to reboot for these changes to take effect. Once that’s done, our hostnames should be reflected in our bash terminal.

|  |  |
| --- | --- |
|  |  |

On the server, we’ll install NFS server. To do so, make sure your repos are updated via sudo apt update and then execute the following command on your server VM:

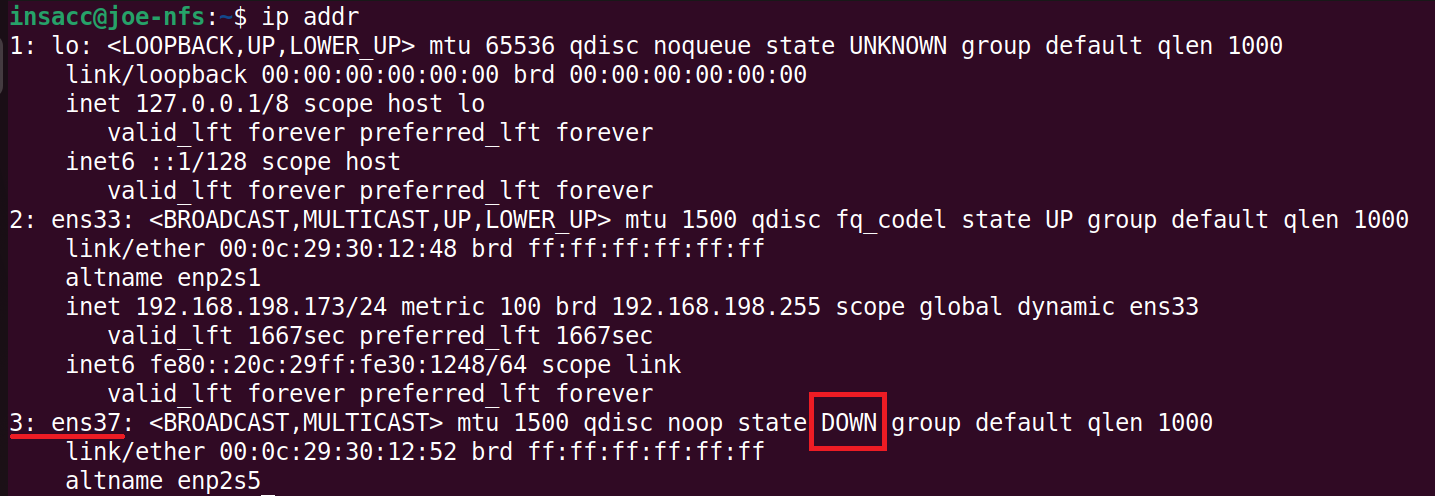
sudo apt install nfs-kernel-server

On the client, install the following after performing an apt update:

sudo apt install nfs-common

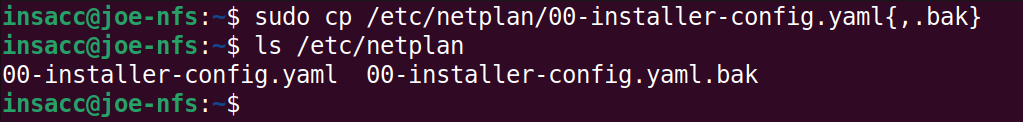
Next, let’s configure our second network adapter that we added to the VM.

Run the command “ip addr” to show IP information:



Make note of your network adapter that doesn’t have an IPv4 address. Let’s assign a static IP to each of our cloned VMs

To assign a static ip address in Ubuntu 20.14, we need to utilize something called netplan. In older versions of this distribution, we could edit the files /etc/network/interfaces. In our distro, we’ll need to edit the file /etc/netplan/00-installer-config.yaml. First, let’s backup that file:



Inside this file:

Text

Description automatically generated

Notice we have **ens33** (our NAT’d adapter). It receives an IP from a DHCP server.

Our new adapter, **ens37**, is not present, let’s add the following:

Text

Description automatically generated

* Once saved, we’ll need to apply these changes via the following command: **sudo netplan apply**

Text

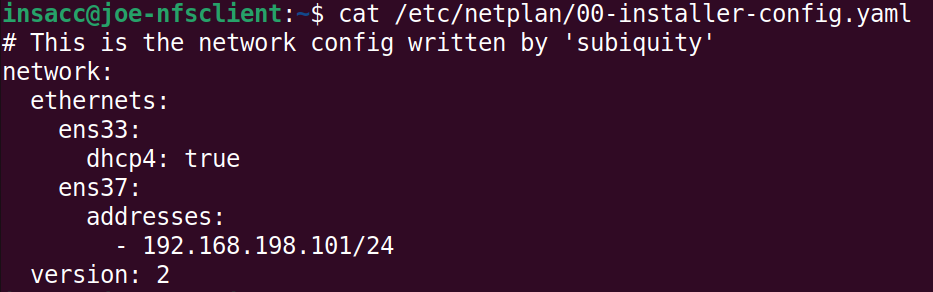
Description automatically generated

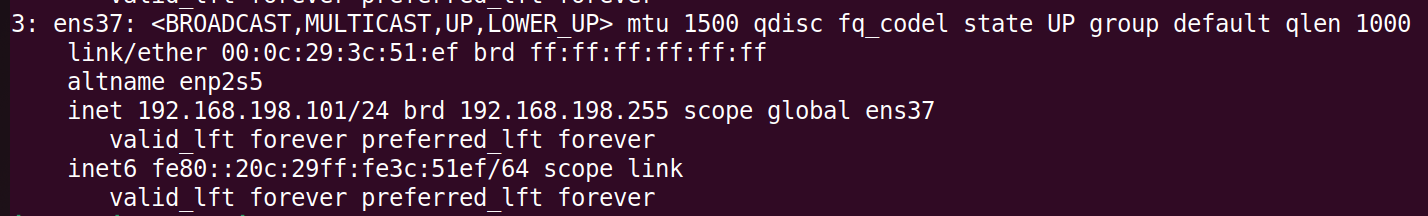
Run the command “ip addr” to confirm:

Text

Description automatically generated

Let’s do the same thing on our Client clone, but use the ip address of 192.168.198.101/24





We now have two cloned VMs, each with a second network adapter with a static IP:

* 192.168.198.100 (Server)
* 192.168.198.101 (Client)

Let’s check if we can ping them from each other.

|  |  |
| --- | --- |
|  |  |

We should be ready to go to start with NFS.

On our server, we’ll need to create a directory, put a file(s) in it, maybe change the ownership of that directory, edit some config files, maybe a restart, and that’s about it.

Let’s create a directory in our /srv/ directory, and we can call it “demoshare”:

Text

Description automatically generated

Since we created that directory with sudo, notice the owner permissions. It’s root, and we don’t want that. We’ll change the directory ownership to nobody:nogroup. To do that, we’ll use the command:

* sudo chown nobody:nogroup /srv/demoshare

Text

Description automatically generated

Next, we need to tell NFS to export that directory and share it. We will need to edit the file **/etc/exports** (make sure to create a backup first).

Text

Description automatically generated

We’ll add a line to the end. This line basically tells Linux to share a path to a certain client with certain properties:

Text

Description automatically generated

In this file, you will list the names of the directories to be exported, the computers that will be allowed to access the directory, and the exporting options. For example, to make a directory named /srv/demoshare available to a computer named **joe-nfsclient**, we would add the line:

* /srv/demoshare joe-nfsclient(rw,sync,no\_subtree\_check)

The options are found between the parentheses. Those listed are the commonly-used options. You can read the exports manpage (man 5 exports) to see all the options. These ones are:

* rw: allows clients to both read from and write to the filesystem
* sync: makes writes synchronous. We should always specify sync unless you have a specific use case not to.
* no\_subtree\_check: This improves reliability. Read the manpage for details:

Some other options you may need:

* ro: the filesystem is exported read-only
* root\_squash: any results from the root user or root group is converted into being from the nobody user or nogroup group. This keeps root on one machine from being root on your filesystem. This is set by default, so you don’t usually specify it.
* no\_root\_squash: Doesn’t do the root\_squash. If you’re going to do this, you better be sure it’s what you want to do!

After we have updated our /etc/exports file, we need to tell NFS to re-read it. We can use the exportfs -a command or simply restart nfs:

Text

Description automatically generated with medium confidence

We are done on the server end. Let’s hook the client into it. We’ll need to create a mount directory, mount it, and test it.

Let’s create a directory inside /mnt called nfs1:

Text

Description automatically generated

Next, we’ll mount the NFS share from our server. To do this, we need to specify the server location in our mount command:

* sudo mount -t nfs 192.168.198.100:/srv/demoshare /mnt/nfs1

Text

Description automatically generated

Create file and add some text. Then confirm that it appears on the server:

Text

Description automatically generated

Text

Description automatically generated

## Practice:

Create a directory under your home directory named nfs-export. Create a file there named can.you.see.me. Make this directory available to another student using the usual export options.

Have another student that you have not exported your filesystem to try to your mount your filesystem. They should receive an error message. Update your system to export to that other student. Have them try again. It should work this time.

Have both students unmount your filesystem. Then change your system to export read-write to one student, and read-only to the other. Have students try mounting and try writing to the filesystem. The rw student should be able to create a file; the ro student should not.