Module 8 - Storage & Networking

By Juan Medina

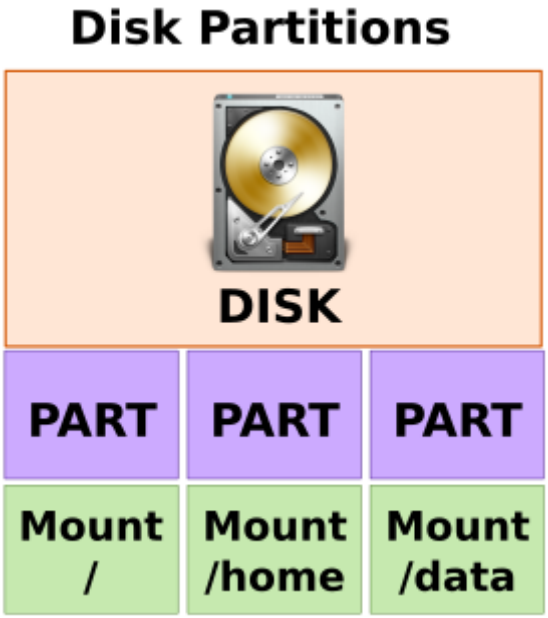
jmedina@collin.edu

COLLIN COLLEGE

Storage

Storage management looks a lot like a giant set of Lego blocks that you can put together in an infinite variety of configurations. Traditional hard disks remain the dominant medium for storage, but they're increasingly being joined by solid-state drives (`SSDs`) for performance-sensitive applications. On top of this hardware is a variety of software components like drivers,partitioning, RAID implementations, Logical Volume Managers, network storage and virtualization that mediate between the raw storage devices and the file-system hierarchy seen by users.

Disk Partitions



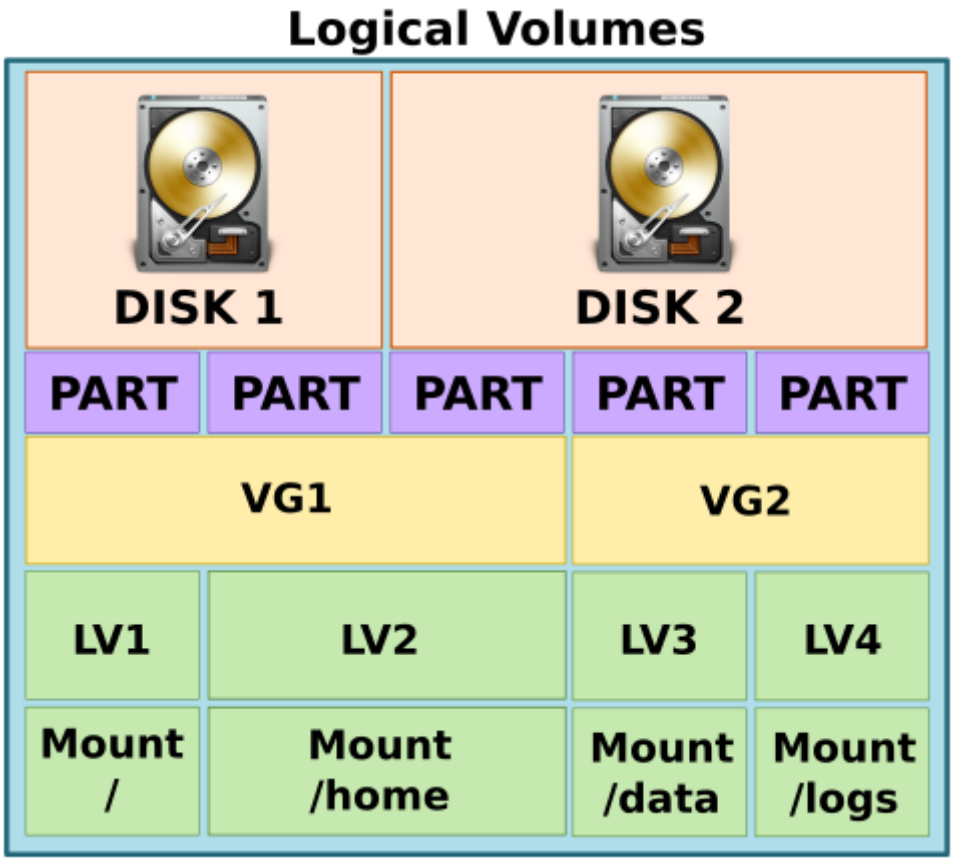
Storage devices are typically divided up into smaller chunks called partitions. Partitions allow you to organize a disk. The various partitions can be formatted with different file systems or used for different purposes. For example, one partition can contain user home directories while another can contain system data and logs.

Partitions are block devices in their own right.

A long listing of the `/dev/sd*` directory reveals its special file type as `b`, which stands for block device:

```
[jmedinar@localhost ~]$ ls -l /dev/sd*
brw-rw----. 1 root disk 8,  0 Dec 28 11:01 /dev/sda
brw-rw----. 1 root disk 8,  1 Dec 28 11:01 /dev/sda1
brw-rw----. 1 root disk 8,  2 Dec 28 11:01 /dev/sda2
brw-rw----. 1 root disk 8, 16 Dec 28 11:01 /dev/sdb
brw-rw----. 1 root disk 8, 17 Dec 28 11:01 /dev/sdb1
brw-rw----. 1 root disk 8, 32 Dec 28 11:01 /dev/sdc
brw-rw----. 1 root disk 8, 33 Dec 28 11:01 /dev/sdc1
brw-rw----. 1 root disk 8, 34 Dec 28 17:32 /dev/sdc2
```

Logical Volumes



Another way of organizing disks and partitions is with logical volume management (LVM). With LVM, one or more block devices can be aggregated into a storage pool called a volume group. Disk space in the volume group is then parceled out to one or more logical volumes, which are the functional equivalent of a partition residing on a physical disk.

File systems

Files on a Linux server are accessed through the file-system hierarchy, a single inverted tree of directories. Each file system is a storage device that has been formatted to store files.

In a sense, the Linux file system hierarchy presents a collection of file systems on separate storage devices as if it were one set of files on one giant storage device that you can navigate.

File systems must be implemented in terms of raw disk blocks. Linux developed a well-defined kernel interface that allowed multiple types of file-systems to be active at once. A good file-system usually provides:

- Good performance
- Tolerance for crashes and power outages without file-system corruption
- The ability to handle disks and files large enough for your needs

Fortunately, Linux default file system cover all of these.

The "fourth extended file system, `ext4`, is the mainstream Linux standard. With journaling capability that increases reliability enormously. If a crash occurs, the file system uses the journal log to reconstruct a perfectly consistent file system. This also reduces the time needed to perform file system consistency checks to approximately one second per file system. `ext4` also raises size limits, increases the performance of certain operations, and allows the use of "extents" (disk block ranges) for storage allocation rather than just individual disk blocks.

Mount Points

To make the contents of a file system available in the file system hierarchy, it must be mounted on an empty directory. This directory is called a mount point. Once mounted, if you use `ls` to list that directory, you will see the contents of the mounted file system, and you can access and use those files normally. Many file systems are automatically mounted as part of the boot process.

Block Devices

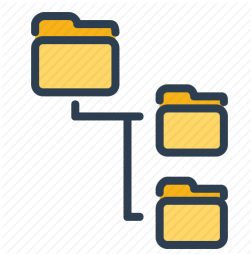
Low-level access to storage devices in Linux is provided by a special type of file called a **block device** These block devices must be formatted with a file system before they can be mounted.

Block device files are stored in the `/dev` directory, along with other device files. Device files are created automatically by the operating system. In Linux, the first SATA/PATA, SAS, SCSI, or USB hard drive detected is called `/dev/sda`, the second is `/dev/sdb`, and so on.

These names represent the entire hard drive. Other types of storage will have other forms of naming.

TYPE OF DEVICE	DEVICE NAMING PATTERN
SATA/SAS/USB-attached storage	/dev/sd*
virtio-blk paravirtualized storage	/dev/vd*
NVMe-attached storage (many SSDs)	/dev/nvme*
SD/MMC/eMMC storage (SD cards)	/dev/mmcblk*
Floppy disk drives.	/dev/fd*

TYPE OF DEVICE	DEVICE NAMING PATTERN
IDE Storage	/dev/hd*
Printers	/dev/lp*
Optical drives (CD/DVD readers and burners)	/dev/sr*
USB Drives	/dev/usb*
Virtual Console Terminal	/dev/vcs*
Terminals (Tele-Typewriter)	/dev/tty*
Device Mapper from LVM	/dev/dm*



Creating file systems

Let's say that we want to reformat a flash drive with a Linux native file system, rather than the FAT32 system they normally have by default. This involves two steps:

1. Create a new **partition** layout (optional).
2. Create a new **file system** on the drive.

Warning! In the following exercise, we are going to format a flash drive. Use a drive that contains nothing you care about because it will be erased! Again, make absolutely sure you are specifying the correct device name for your system, not the one shown in this example.

Identify the proper disk

Let's review the mounted disks in our system

```
[root@localhost ~]# fdisk -l | grep 'Disk /' | sort
Partition 1 does not start on physical sector boundary.
Disk /dev/loop0: 96.98 MiB, 101695488 bytes, 198624 sectors
Disk /dev/loop1: 8 KiB, 8192 bytes, 16 sectors
Disk /dev/mapper/fedora_localhost--live-home: 387.02 GiB,
415563251712 bytes, 811646976 sectors
Disk /dev/mapper/fedora_localhost--live-root: 70 GiB, 75161927680
bytes, 146800640 sectors
Disk /dev/mapper/fedora_localhost--live-swap: 7.74 GiB, 8308916224
bytes, 16228352 sectors
Disk /dev/sda: 167.68 GiB, 180045766656 bytes, 351651888 sectors
Disk /dev/sdb: 298.09 GiB, 320072933376 bytes, 625142448 sectors
Disk /dev/sdc: 5.46 TiB, 6001175125504 bytes, 11721045167 sectors
Disk /dev/sdd: 7.5 GiB, 8054112256 bytes, 15730688 sectors
Disk /dev/zram0: 4 GiB, 4294967296 bytes, 1048576 sectors
```

I used `grep` to reduce our output just enough to see the Disk names and sorted the output so I can easily identify which one is my inserted usb drive

I can by default ignore `mapper` drives because I know those represent `LVM` not USB.

I can also ignore `loop` because those are different devices again not the USB.

That leave me with just a few options:

```
Disk /dev/sda: 167.68 GiB, 180045766656 bytes, 351651888 sectors
Disk /dev/sdb: 298.09 GiB, 320072933376 bytes, 625142448 sectors
Disk /dev/sdc: 5.46 TiB, 6001175125504 bytes, 11721045167 sectors
Disk /dev/sdd: 7.5 GiB, 8054112256 bytes, 15730688 sectors
Disk /dev/zram0: 4 GiB, 4294967296 bytes, 1048576 sectors
```

I can ignore the ones that doesn't match the size of my USB drive and that leaves me with the proper answer

```
Disk /dev/sdd: 7.5 GiB, 8054112256 bytes, 15730688 sectors
```

I can double check with a couple commands.

First with `mount` that actually shows me the brand of my USB `maxell`

```
[root@localhost ~]# mount | grep sdd
/dev/sdd1 on /run/media/jmedinar/maxell type vfat
(rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=0022,dmask=0022,co
depage=437,iocharset=ascii,shortname=mixed,showexec,utf8,flush,error
s=remount-ro,uhelper=udisks2)
```

Or with the `lsblk` command

```
[root@localhost ~]# lsblk | grep sdd
sdd                             8:48   1   7.5G  0 disk
└─sdd1                          8:49   1   7.5G  0 part
/run/media/jmedinar/maxell
```

Manipulating partitions with fdisk

The `fdisk` program allows us to interact directly with disk-like devices at a very low level. With this tool, we can edit, delete, and create partitions on the device. To work with our flash drive, we must first unmount it (if needed) and then invoke the `fdisk` program as follows:

```
[root@localhost ~]# umount /run/media/jmedinar/maxell
[root@localhost ~]# fdisk /dev/sdd
```

We will be presented with a series of screens and multiple options. Read carefully!

```
[root@localhost ~]# fdisk /dev/sdd

Welcome to fdisk (util-linux 2.36).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help):
```

We select the `m` option, notice that we must specify the device in terms of the entire device, not by partition number. :

```
Command (m for help): m

Help:

  DOS (MBR)
   a   toggle a bootable flag
   b   edit nested BSD disklabel
   c   toggle the dos compatibility flag

  Generic
   d   delete a partition
   F   list free unpartitioned space
   l   list known partition types
   n   add a new partition
   p   print the partition table
   t   change a partition type
   v   verify the partition table
   i   print information about a partition

  Misc
   m   print this menu
   u   change display/entry units
   x   extra functionality (experts only)

  Script
   I   load disk layout from sfdisk script file
   O   dump disk layout to sfdisk script file

  Save & Exit
   w   write table to disk and exit
   q   quit without saving changes

  Create a new label
   g   create a new empty GPT partition table
   G   create a new empty SGI (IRIX) partition table
   o   create a new empty DOS partition table
   s   create a new empty Sun partition table
```

The first thing we want to do is examine the existing partition layout. We do this by entering `p` to print the partition table for the device:

```
Command (m for help): p
Disk /dev/sdd: 7.5 GiB, 8054112256 bytes, 15730688 sectors
Disk model: USB Disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xb70b8e46

Device     Boot Start      End  Sectors  Size Id Type
/dev/sdd1        2320 15730687 15728368  7.5G  c W95 FAT32 (LBA)
```

In this example, we see a `7.5` GB device with a single partition `/dev/sdd1`. The partition is identified as a `Windows 95 FAT32` partition. Some programs will use this identifier to limit the kinds of operations that can be done to the disk, but most of the time it is not critical to change it. We will change it to indicate a Linux partition.

We must first find out what ID is used to identify a Linux partition using the option `l`.

```
Command (m for help): l

00 Empty            24 NEC DOS          81 Minix / old Lin  bf
Solaris
01 FAT12            27 Hidden NTFS Win  82 Linux swap / So  c1
DRDOS/sec (FAT-
02 XENIX root       39 Plan 9           83 Linux            c4
DRDOS/sec (FAT-
03 XENIX usr        3c PartitionMagic   84 OS/2 hidden or   c6
DRDOS/sec (FAT-
04 FAT16 <32M       40 Venix 80286      85 Linux extended   c7
Syrinx
05 Extended         41 PPC PReP Boot    86 NTFS volume set  da Non-
FS data
06 FAT16            42 SFS              87 NTFS volume set  db CP/M
/ CTOS / .
07 HPFS/NTFS/exFAT  4d QNX4.x           88 Linux plaintext  de Dell
Utility
08 AIX              4e QNX4.x 2nd part  8e Linux LVM        df
BootIt
09 AIX bootable     4f QNX4.x 3rd part  93 Amoeba           e1 DOS
access
0a OS/2 Boot Manag  50 OnTrack DM       94 Amoeba BBT       e3 DOS
R/O
0b W95 FAT32        51 OnTrack DM6 Aux  9f BSD/OS           e4
SpeedStor
0c W95 FAT32 (LBA)  52 CP/M             a0 IBM Thinkpad hi  ea
Linux extended
0e W95 FAT16 (LBA)  53 OnTrack DM6 Aux  a5 FreeBSD          eb BeOS
fs
0f W95 Ext'd (LBA)  54 OnTrackDM6       a6 OpenBSD          ee GPT
```

```
10 OPUS              55 EZ-Drive         a7 NeXTSTEP         ef EFI
(FAT-12/16/
11 Hidden FAT12      56 Golden Bow       a8 Darwin UFS       f0
Linux/PA-RISC b
12 Compaq diagnost   5c Priam Edisk      a9 NetBSD           f1
SpeedStor
14 Hidden FAT16 <3   61 SpeedStor        ab Darwin boot      f4
SpeedStor
16 Hidden FAT16      63 GNU HURD or Sys  af HFS / HFS+       f2 DOS
secondary
17 Hidden HPFS/NTF   64 Novell Netware   b7 BSDI fs          fb
VMware VMFS
18 AST SmartSleep    65 Novell Netware   b8 BSDI swap        fc
VMware VMKCORE
1b Hidden W95 FAT3   70 DiskSecure Mult  bb Boot Wizard hid  fd
Linux raid auto
1c Hidden W95 FAT3   75 PC/IX            bc Acronis FAT32 L  fe
LANstep
1e Hidden W95 FAT1   80 Old Minix        be Solaris boot     ff BBT


Aliases:
   linux          - 83
   swap           - 82
   extended       - 05
   uefi           - EF
   raid           - FD
   lvm            - 8E
   linuxex        - 85

```

In the list we can see the `83` for a regular Linux partition or the option `8e` for Linux LVM. To change it we use the option `t`.

```
Command (m for help): t
Selected partition 1
Hex code or alias (type L to list all): 83
Changed type of partition 'W95 FAT32 (LBA)' to 'Linux'.
```

This completes all the changes that we need to make. Up to this point, the device has **been untouched** (all the changes have been stored in memory, not on the physical device), so we will write the modified partition table to the device and exit. To do this, we enter `w` at the prompt.

```
Command (m for help): w
The partition table has been altered!
```

If we had decided to leave the device unaltered, we could have entered `q` at the prompt to quit.

Creating a file system with mkfs

With our partition editing done it's time to create a new file system on our flash drive. To do this, we will use `mkfs` command, that can create file systems in a variety of formats. To create an `ext4` file system on the device, we use the `-t` option to specify the type `ext4`, followed by the name of the device containing the partition we wish to format:

```
[root@localhost ~]# mkfs -t ext4 /dev/sdd1
```

The program will display a lot of information when `ext4` is the chosen file system type. To re-format the device to its original `FAT32` file system, specify `vfat` as the file system type:

```
[root@localhost ~]# mkfs -t vfat /dev/sdd1
```

This process of partitioning and formatting can be used anytime additional storage devices are added to the system.

Mounting

A file system residing on a removable storage device needs to be mounted in order to access it. The mount command allows the root user to manually mount a file system. The first argument of the mount command specifies the file system to mount. The second argument specifies the directory to use as the mount point in the file-system hierarchy.

There are two common ways to specify the file system on a disk partition to the mount command:

- With the name of the device file in `/dev` containing the file system.
- With the `UUID` written to the file system, a universally-unique identifier.

Mounting a device is relatively simple. You need to identify the device you want to mount, make sure the mount point exists, and mount the device on the mount point.

Identifying the Block Device

A hot-pluggable storage device, whether a hard disk drive `HDD` or solid-state device `SSD` in a server caddy, or a `USB` storage device, might be plugged into a different port each time they are attached to a system.

Use the `lsblk` command to list the details of a specified block device or all the available devices.

```
[root@localhost ~]# lsblk
NAME                          MAJ:MIN RM   SIZE RO TYPE
MOUNTPOINT
loop0                           7:0    0    97M  1 loop
/var/lib/snapd/snap/core/9665
loop1                           7:1    0     8K  1 loop
/var/lib/snapd/snap/openboard/14
sda                             8:0    0 167.7G  0 disk
```

```
├─sda1                            8:1    0     1G  0 part /boot
└─sda2                            8:2    0 166.7G  0 part
  ├─fedora_localhost--live-root 253:0    0    70G  0 lvm  /
  ├─fedora_localhost--live-swap 253:1    0   7.7G  0 lvm  [SWAP]
  └─fedora_localhost--live-home 253:2    0   387G  0 lvm  /home
sdb                               8:16   0 298.1G  0 disk
└─sdb1                            8:17   0 298.1G  0 part
  └─fedora_localhost--live-home 253:2    0   387G  0 lvm  /home
sdc                               8:32   0   5.5T  0 disk
├─sdc1                            8:33   0   128M  0 part
└─sdc2                            8:34   0   5.5T  0 part
/run/media/jmedinar/Seagate Backup Plus Drive
sdd                               8:48   1   7.5G  0 disk
└─sdd1                            8:49   1   7.5G  0 part
```

If you know that you just added a `7.5G` storage device with one partition, then you can guess from the preceding output that `/dev/sdd1` is the partition that you want to mount.

Mounting by Block Device Name

The following example mounts the file system in the`/dev/sdd1` partition on the directory `/mnt/data`.

```
[root@localhost ~]# mount /dev/sdd1 /mnt/data
```

To mount a file system, the destination directory must already exist.

> *IMPORTANT*
>
> *If the directory acting as mount point is not empty, any files copied to that directory before the file system was mounted are not accessible until the file system is unmounted again.*

This approach works fine in the short run. However, the order in which the operating system detects disks can change if devices are added to or removed from the system. This will change the device name associated with that storage device. A better approach would be to mount by some characteristic built into the file system.

Mounting by File-system `UUID`

One stable identifier that is associated with a file system is its `UUID`, a very long hexadecimal number that acts as a universally-unique identifier. This `UUID` is part of the file system and remains the same as long as the file system is not recreated.

The `lsblk -fp` command lists the full path of the device, along with the `UUIDs` and mount points, as well as the type of file system in the partition. If the file system is not mounted, the mount point will be blank.

```
[root@localhost ~]# lsblk -fp /dev/sda
NAME                                  FSTYPE      FSVER    LABEL UUID                                   FSAVAIL FSUSE% MOUNTPOINT
/dev/sda
├─/dev/sda1                           ext4        1.0            99dcd15a-4811-43dc-b39c-c6812de80a83    636.7M    28% /boot
└─/dev/sda2                           LVM2_member LVM2 001       XxDjmF-W1vl-q4fd-LUFK-Xg4I-aIIQ-cHoslZ
  ├─/dev/mapper/fedora_localhost--live-root ext4  1.0            abf65797-144a-4588-ade5-17eec9b283f5     41.3G    35% /
  ├─/dev/mapper/fedora_localhost--live-swap swap  1              6dd9deaa-310f-4216-a675-be18a0928f7e                  [SWAP]
  └─/dev/mapper/fedora_localhost--live-home ext4  1.0            e7a61212-08c2-4dfa-bfc1-b39dbc51d50c    312.6G    13% /home
```

Mount the file system by the UUID of the file system.

```
[root@localhost ~]# mount UUID="e7a61212-08c2-4dfa-bfc1-b39dbc51d50c" /mnt/data
```

Automatic mounting

If you are logged in and using the graphical desktop environment, it will automatically mount any removable storage media when it is inserted. The removable storage device is mounted at `/run/media/USERNAME/LABEL` where LABEL is an identifier, often the name given to the file system when it was created if one is available.

Before removing the device, you should unmount it manually.

If you are not using a removable device but a permanent one you have to include it into the `/dev/fstab` configuration to have automatically mounted at boot time.

The `fstab` is your operating system's file system table. The `fstab` is configured to look for specific file systems and mount them automatically in a desired way each and every time.

```
#
# /etc/fstab
# Created by anaconda on Thu Feb 27 19:31:06 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/fedora_localhost--live-root        /       ext4    defaults        1 1
UUID=99dcd15a-4811-43dc-b39c-c6812de80a83      /boot   ext4    defaults        1 2
/dev/mapper/fedora_localhost--live-home        /home   ext4    defaults        1 2
/dev/mapper/fedora_localhost--live-swap        none    swap    defaults        0 0
```

The `fstab` uses the same mechanism as the `mount` command so we can mount by `partition` or `uuid`. Let's review the fields in the file:

```
1 /dev/mapper/fedora_localhost--live-root       /
    ext4    defaults        1 1
```

1. device path or uuid to mount
2. mount point
3. file system type
4. mounting options
5. dumping flag
6. passing flag

Mounting options

There's a large set of options available, but there's a handful or so of very common ones.

Option	Description
auto/noauto	Specify whether the partition should be automatically mounted on boot. You can block specific partitions from mounting at boot-up by using "noauto".
exec/noexec	Specifies whether the partition can execute binaries. If you have a scratch partition that you compile on, then this would be useful, or maybe if you have /home on a separate file system. If you're concerned about security, change this to "noexec".
ro/rw	`ro` is read-only, and `rw` is read-write. If you want to be able to write to a file-system as the user and not as root, you'll need to have `rw` specified.
sync/async	This one is interesting. "sync" forces writing to occur immediately on execution of the command, which is ideal for floppies and USB drives, but isn't entirely necessary for internal hard disks. What "async" does is allow the command to execute over an elapsed time period, perhaps when user activity dies down and the like. Ever get a message asking to your "wait while changes are being written to the drive?" This is usually why.
nouser/user	This allows the user to have mounting and unmounting privileges. An important note is that `user` automatically implies `noexec` so if you need to execute binaries and still mount as a user, be sure to explicitly use `exec` as an option.

These options are separated by a comma and no spaces, and can be put in any order. If you're not sure about the defaults, it's okay to explicitly state your options. Things that are mounted from temporary places (like USB) won't follow this basic pattern unless you created entries for them (by UUID) in fstab.

dumping

The next option is a binary value `0` for false and `1` for true for "dumping." This is a pretty much out-dated method of backup for cases when the system went down. You should leave this as `0`.

passing

The last option is a numeric value for "passing." This tells the system the order in which to perform a file system check `fsck`.

- `0` it will be skipped
- `1-n` it will be checked in that order

The root file system should always be `1` and other file systems can go afterward. This works best for journaling file systems like `ext3/4` and `ReiserFS`. Older file systems like `FAT16/32` and `ext2` can take a while, so it's better to turn their fscking off and do it periodically yourself.

Unmounting

The shutdown and reboot procedures unmount all file systems automatically. As part of this process, any file system data cached in memory is flushed to the storage device thus ensuring that the file system suffers no data corruption.

To unmount a file system, the `umount` command expects the mount point as an argument.

```
[root@localhost ~]# umount /mnt/data
```

Unmounting is not possible if the mounted file system is in use. For the `umount` command to succeed, all processes needs to stop accessing data under the mount point.

```
[root@localhost ~]# cd /mnt/data
[root@localhost data]# umount /mnt/data
umount: /mnt/data: target is busy.
```

The `lsof` command lists all open files and the process accessing them in the provided directory. It is useful to identify which processes currently prevent the file system from successful unmount.

```
[root@localhost data]# lsof /mnt/data
COMMAND  PID   USER  FD    TYPE  DEVICE   SIZE/OFF NODE  NAME
bash     1593  root  cwd   DIR   253,17   6        128   /mnt/data
lsof     2532  root  cwd   DIR   253,17   19       128   /mnt/data
lsof     2533  root  cwd   DIR   253,17   19       128   /mnt/data
```

Once the processes are identified, an action can be taken, such as waiting for the process to complete or sending a SIGTERM or SIGKILL signal to the process.

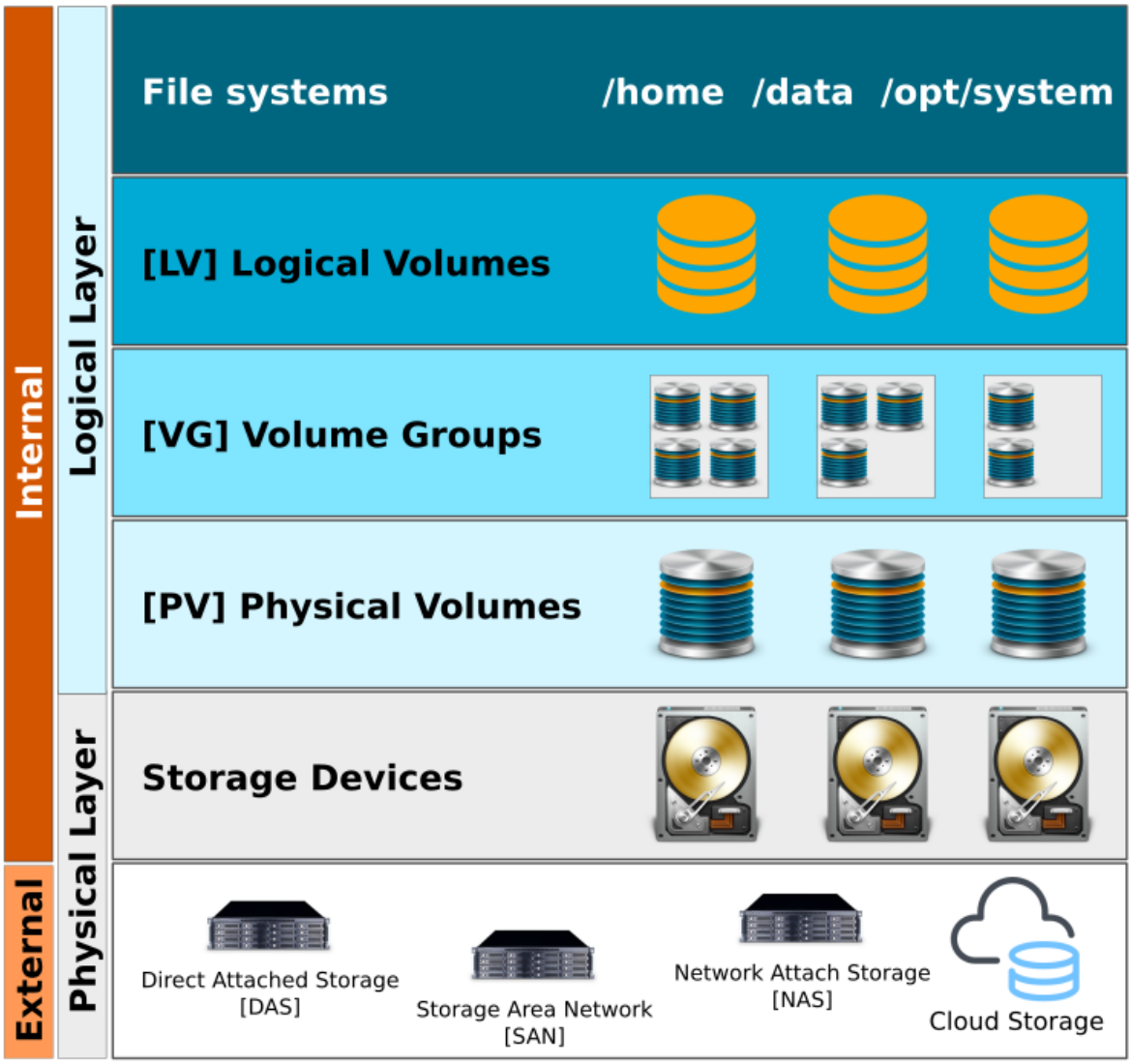
```
[root@host ~]# umount /mnt/data
```

Logical Volume Management

Logical Volume Management `LVM` creates a layer of abstraction over physical storage, allowing you to create logical storage volumes. With `LVM` in place, you are not bothered with physical disk sizes because the hardware is hidden from the software so it can be resized and moved without stopping applications or unmounting file systems. You can think of LVM as dynamic partitions.

Advantages:

- **Resize storage pools:** You can extend the logical space as well as reduce it without reformatting the disks.
- **Flexible storage capacity:** You can add more space by adding more disks and adding them to the pool of physical storage, thus you have flexible storage capacity.



Requesting new storage

Requesting storage in the server's world is normally something that happens outside of the server itself, you can as some might imagine open the server box and insert the new disk connecting it to cables and that will be consider as `internal storage` but that is very limiting to the number of slots in the server to connect storage and space not to say time consuming, risky and energy consuming in reality in most cases the request will be from some sort of external storage device [DAS, SAN, NAS, Cloud] that is already connected or accessible to the server.

As system administrator most of the time you won't be in charge of the storage itself so you should only care about the storage once it has been attached to the server and you can start using it. We refer most commonly to any type of storage as "LUN" Logical Unit Number, block device or simply "Disk".

Behind the scenes a lot of things might have happened you might be getting RAID storage, SATA, ATA, SCSI, SDD, it might have been connected over the network, serial or Fibre, it might be 10GbE connection or 1-GbE it might have redundancy or not, and high availability or not, etc etc...

But at the end all you care is about your LUN being visible to your Linux server!

Storage devices

Once the LUN is attached to your server we have to format it using `fdisk` command we add the new disk to be usable by LVM (8e).

First identify the disks available in the system and the one you will be adding

```
[root@localhost ~]# fdisk -l | grep Disk
```

Double-check is not an already assigned disk

```
[root@localhost ~]# lsblk
[root@localhost ~]# fdisk -c -u /dev/sdd
```

Start the `fdisk` program against the disk you want to format:

```
[root@localhost ~]# fdisk /dev/sdd
```

Select the following options in the screens:

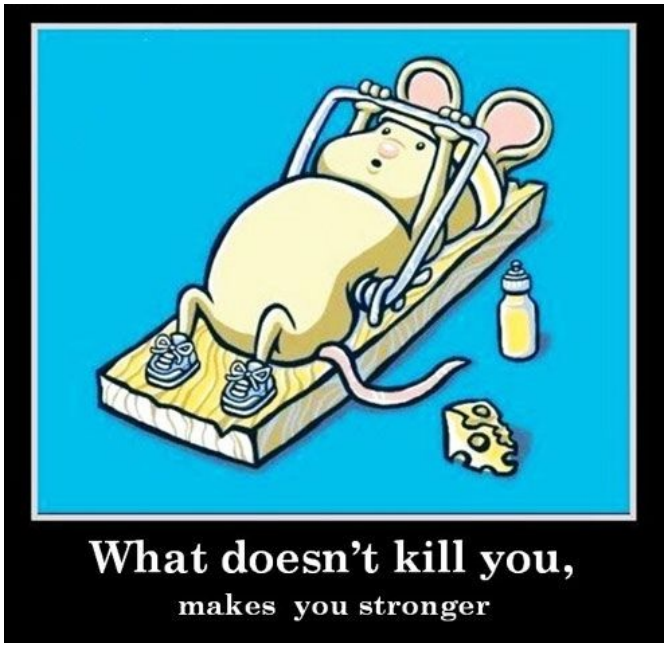
- n: add a new partition
- p: primary partition
- For partition number just hit Enter to select the default
- First Sector just hit Enter to select the default
- Last Sector just hit Enter to select the default
- t: to change the partition type
- Set 8e as code

Now let's review our new partition table

- p: to print the partition table

Finally, let's write the changes

- w: to write the changes



Done. We are done with all the physical activity! Let's get into the Logical! LVM!

Physical Volumes

A physical volume is the actual storage device **representation** that will be used in the LVM configuration. It can be an entire disk or a partition of a disk.

PV Commands:

Command	Description
pvcreate	Initialize physical volume(s) for use by LVM
pvdisplay	Display various attributes of physical volume(s)
pvremove	Remove LVM label(s) from physical volume(s)
pvchange	Change attributes of physical volume(s)
pvmove	Move extents from one physical volume to another
pvresize	Resize physical volume(s)
pvscan	List all physical volumes
pvck	Check metadata on physical volumes
pvs	Display information about physical volumes

The **pvcreate** command initializes the disk so it can be a part of future volume group.

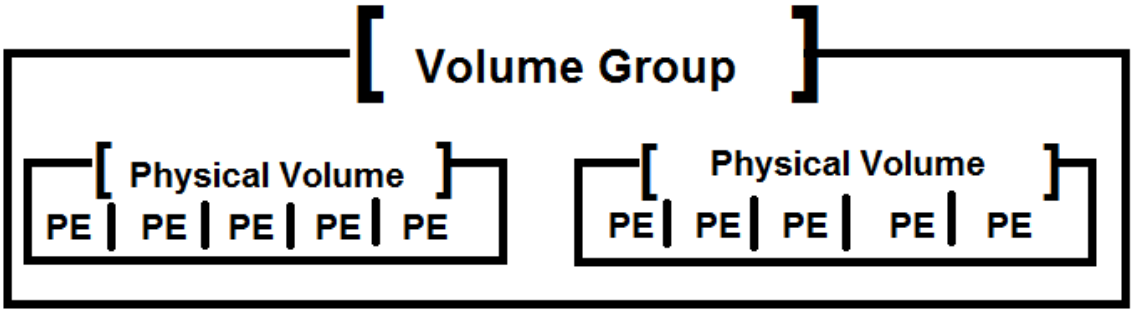
```
[root@localhost ~]# pvcreate /dev/sdb
```

You can display PVs with the pvdisplay command

```
[root@localhost ~]# pvdisplay
```

Volume Groups

Physical Volumes are combined into Volume Groups `VGs`. To create a pool of disk space out of which logical volumes can be allocated. The disk space available for allocation in Volume Group is divided into units of a fixed-size called extents. An extent is the smallest unit of storage that can be allocated. Within a physical volume, extents are referred to as physical extents `PE`. A logical volume is allocated into logical extents of the same size as the physical extents. The extent size is thus the same for all logical volumes in the volume group. The volume group maps the logical extents to physical extents.



So let's create a VG:

```
[root@localhost ~]# vgcreate dataVG /dev/sdb
```

You can display information

```
[root@localhost ~]# vgdisplay
```

Notice the VG Size and the PE Size those are important as well as the Total PE and Free PE Values.

VG Commands:

Command	Description
vgcreate	Create a volume group
vgdisplay	Display volume group information
vgremove	Remove volume group(s)
vgchange	Change volume group attributes

Command	Description
vgextend	Add physical volumes to a volume group
vgreduce	Remove physical volume(s) from a volume group
vgsplit	Move physical volumes into a new or existing volume group
vgmerge	Merge volume groups
vgexport	Unregister volume group(s) from the system
vgimport	Register exported volume group with system
vgrename	Rename a volume group
vgconvert	Change volume group metadata format
vgscan	Search for all volume group
vgck	Check the consistency of volume group(s)
vgs	Display information about volume groups
vgmknodes	Create the special files for volume group devices in /dev
vgimportclone	Import a VG from cloned PVs
vgcfgbackup	Backup volume group configuration(s)
vgcfgrestore	Restore volume group configuration

Logical Volumes

Once our volumes are grouped we will break them appart again but now to fit our needs into Logical Volumes. Logical Volumes (LV) are the final storage unit in the standard LVM architecture, and we have a few different options to cover unique use cases that we can employ to best fit a given situation.

Linear logical volume

Linear logical volumes are the LVM default when it comes to logical volume creation. They are generally used to combine one or more disks to create one usable storage unit.

Striped logical volume

Striped logical volumes allow the administrator to control the way that the data is written to the physical volumes. For high volume read/write scenarios, striped logical volumes would be ideal, as they allow for read and write operations to be done in parallel.

When using striped logical volumes, you can set the number of stripes (this number cannot exceed the number of physical volumes) and the stripe size. This allows the user a greater level of control over how I/O is performed on the system.

Mirrored logical volume

Mirrored logical volumes do exactly what you would expect them to do. They allow you to "reflect" the data on one device to an identical copy. This ensures your data is available. If one part of the mirror breaks, the remaining drive changes its characteristics to that of a linear volume and is still accessible.

Let's create a Linear logical volume

```
[root@localhost ~]# lvcreate -L 1GB -n mylv myvg
```

You can display information

```
[root@localhost ~]# lvdisplay
```

Notice the LV PAth, LV Status, and LV Size.

LV Commands:

Command	Description
lvcreate	Create a logical volume
lvdisplay	Display information about a logical volume
lvremove	Remove logical volume(s) from the system
lvchange	Change the attributes of logical volume(s)
lvextend	Add space to a logical volume
lvreduce	Reduce the size of a logical volume
lvresize	Resize a logical volume
lvrename	Rename a logical volume
lvconvert	Change logical volume layout
lvscan	List all logical volumes in all volume groups

Command	Description
lvs	Display information about logical volumes

File System

Create a file system from your LV as ext4 so it can be used by Linux

```
[root@localhost ~]# mkfs -t ext4 /dev/vgname/lvname
```

Mounting

Mount the volume to any directory you need to

```
[root@localhost ~]# mkdir /your/mount/point
[root@localhost ~]# mount /dev/mapper/your_lvg /your/mount/point
```

You can also make the mount permanent so it will be available every time you reboot your server.

```
[root@localhost ~]# vim /etc/fstab
```

add a line for your new mount as the following

```
/dev/mapper/fedora_localhost--live-root / ext4 default 0 2
```

LVM partitioning is very useful if you have more than one hard disk, logical volumes can extend and are not limited by the size of one single disk, rather by the total aggregated size.

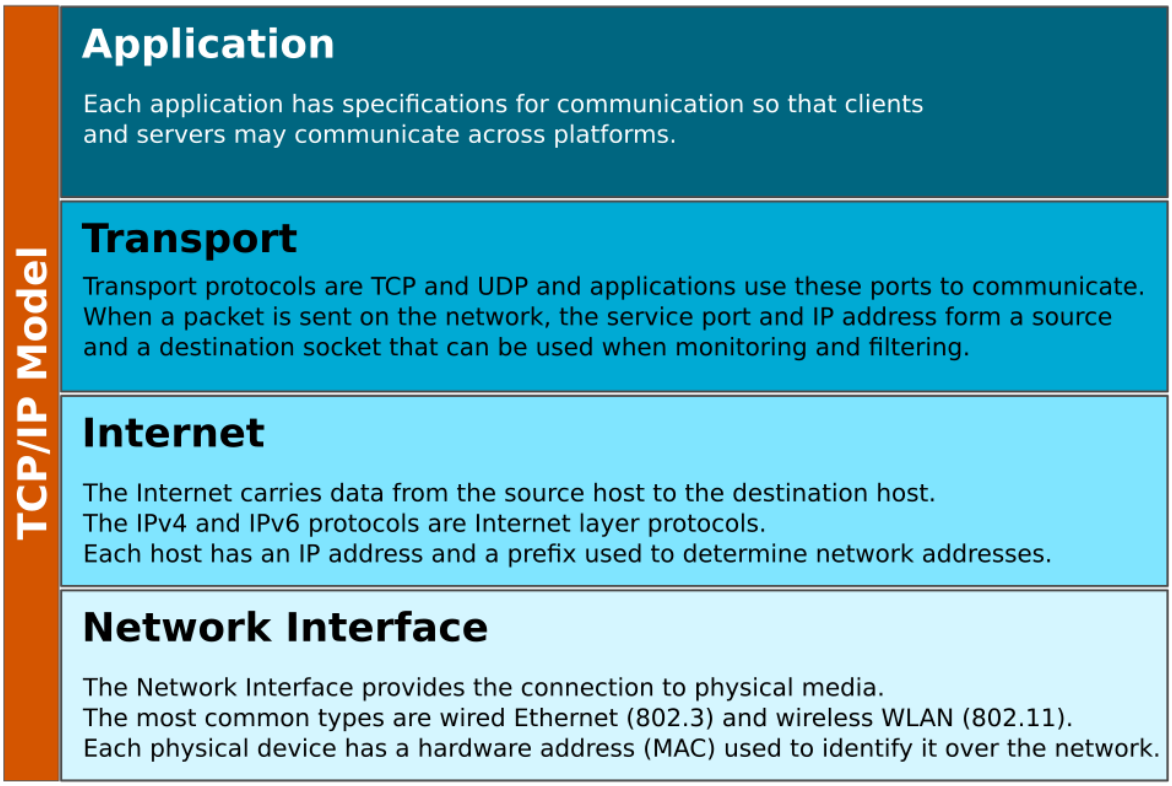
Managing Networking

When it comes to networking, there is probably nothing that cannot be done with Linux. Linux is used to build all sorts of networking systems and appliances, including firewalls, routers, name servers, NAS boxes and on and on. Just as the subject of networking is vast, so are the number of commands that can be used to configure and control it.

Networking concepts

TCP/IP Model

The TCP/IP network model is a simplified, four-layered set of abstractions that describes how different protocols interoperate in order for computers to send traffic from one machine to another over the Internet.



Network interface names

Each network port on a system has a name, which you use to configure and identify it.

Older versions of Linux used names like `eth0`, `eth1`, and `eth2` for each network interface. The name `eth0` was assigned first network port detected by the operating system, eth1 the second, and so on. However, as devices are added and removed, the mechanism detecting devices and naming them could change which interface gets which name.

Newer versions of Linux use a different naming system. Instead of being based on detection order, the names of network interfaces are assigned based on information from the firmware, the PCI bus topology, and type of network device.

Naming system	Description
en	Ethernet Network
wl	[WLAN] Wireless Local Area Network Interface
ww	[WWAN] Wireless Wide Area Network Interface
oN	on-board device with N firmware index number
sN	PCI hotplug in slot N
pMsN	PCI on bus M in slot N
lo	loopback
virbr	Virtual bridge network
docker	docker bridge network

Examples:

```
eno1: Ethernet Network on-board 1
ens3: Ethernet Network PCI hotplug in slot 3
wlp4s0: Wireless PCI on bus 4 in slot 0
virbr0: Virtual Bridge 0
docker0: Docker Bridge 0
```

Persistent naming means that once you know what the name is for a network interface on the system, you also know that it will not change later. The trade off is that you cannot assume that a system with one interface will name that interface `eth0`.

IPv4

An IPv4 address is a 32-bit number, normally expressed in decimal as four 8-bit octets ranging in value from 0 to 255, separated by dots.

The address is divided into two parts:

- The network part (subnet).
- The host part (identifier).

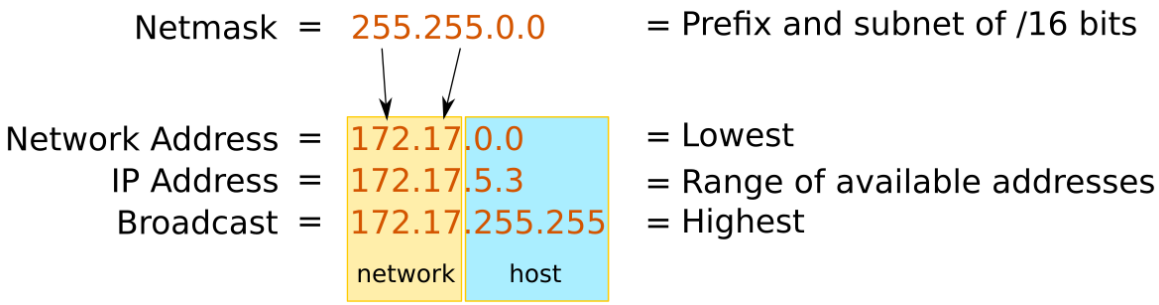
All hosts on the same subnet can talk to each other directly without a router.

No two hosts on the same subnet can have the same host part.

The size of an IPv4 subnet is variable it depends on the netmask, which is assigned to the subnet. The netmask indicates how many bits of the IPv4 address belong to the subnet. The more bits available for the host part, the more hosts can be on the subnet.

The lowest possible address on a subnet is called the network address.

The highest possible address on a subnetis the broadcast address.



The special address 127.0.0.1 always points to the local system ("localhost"), and the network 127.0.0.0/8 belongs to the local system, so that it can talk to itself using network protocols.

IPv6

IPv6 is intended as an eventual replacement for the IPv4 network protocol. You will need to understand how it works since increasing numbers of production systems use IPv6 addressing.

IPv6 can also be used in parallel with IPv4 in a dual-stack model. In this configuration, a network interface can have an IPv6 address or addresses as well as IPv4 addresses. Linux operates in a dual-stack mode by default.

An IPv6 address is a 128-bit number, normally expressed as eight colon-separated groups of four hexadecimal nibbles. Each nibble represents four bits of the IPv6 address, so each group represents 16 bits of the IPv6 address.

```
1 2001:0db8:0000:0010:0000:0000:0000:0001
```

To make IPv6 addresses easier to write, leading zeros in a colon-separated group do not need to be written. However, at least one hexadecimal digit must be written in each colon-separated group.

```
1 2001:db8:0:10:0:0:0:1
```

Since addresses with long strings of zeros are common, one or more consecutive groups of zeros only may be combined with exactly one :: block.

```
1 2001:db8:0:10::1
```

Some tips for writing consistently readable addresses:

- Suppress leading zeros in a group.
- Use :: to shorten as much as possible.

- If an address contains two consecutive groups of zeros, equal in length, it is preferred to shorten the leftmost groups of zeros to :: and the rightmost groups to :0: for each group.
- Although it is allowed, do not use :: to shorten one group of zeros. Use :0: instead, and save :: for consecutive groups of zeros.
- Always use lowercase letters for hexadecimal numbers a through f.

Hostname

It would be inconvenient if you always had to use IP addresses to contact your servers. Humans generally would prefer to work with names than long and hard-to-remember strings of numbers. And so Linux has a number of mechanisms to map a host name to an IP address, collectively called name resolution.

One way is to set a static entry for each name in the `/etc/hosts` file on each system. This requires you to manually update each server's copy of the file.

Another way is using a DNS that is a distributed network of servers providing mappings of host names to IP addresses. In order for name service to work, a host needs to be pointed at a nameserver. This nameserver does not need to be on the same subnet; it just needs to be reachable by the host. This is typically configured through DHCP or a static setting in a file called `/etc/resolv.conf`.

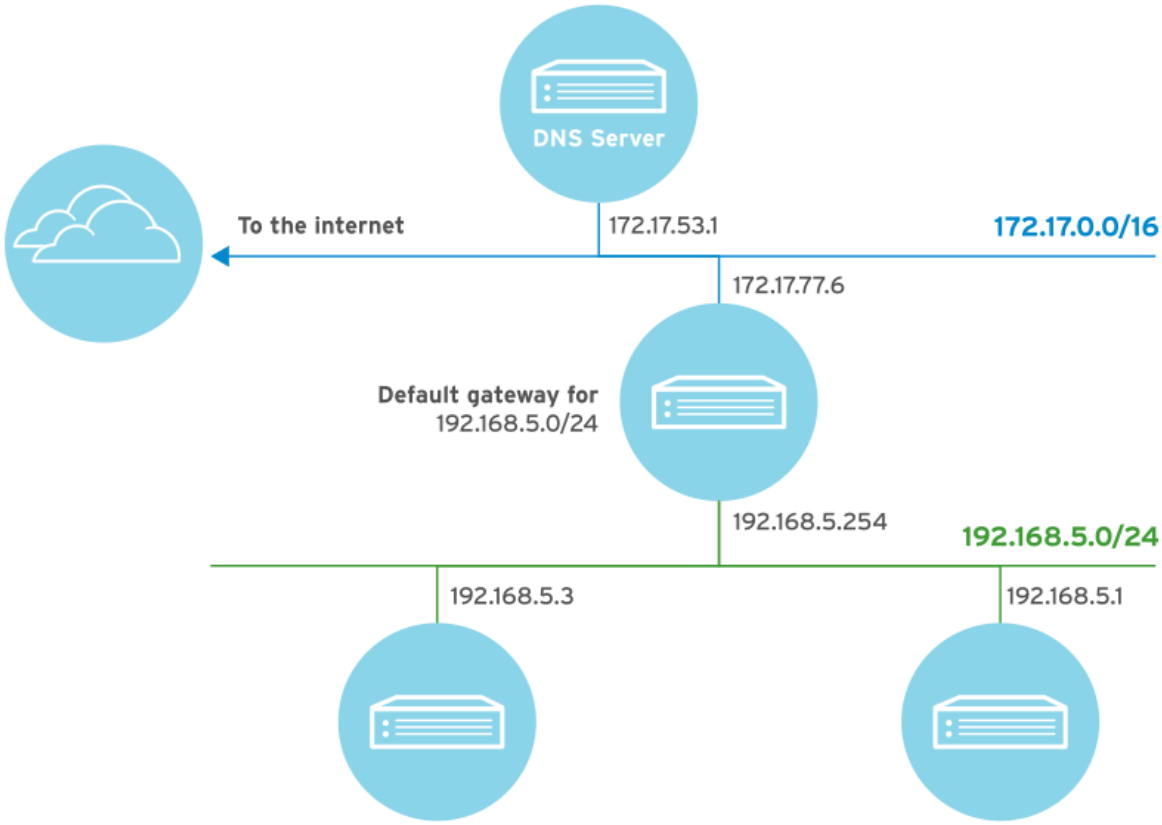
🎗 The Internet: IP Addresses & DNS

Routing

Whether using IPv4 or IPv6, network traffic needs to move from host to host and network to network. Each host has a routing table, which tells it how to route traffic for particular networks. A routing table entry lists a destination network, which interface to use when sending traffic, and the IP address of any intermediate router required to relay a message to its final destination.

If the network traffic does not match a more specific route, the routing table usually has an entry for a default route to the entire IPv4 Internet: 0.0.0.0/0. This default route points to a router on a reachable subnet.

If a router receives traffic that is not addressed to it, instead of ignoring it like a normal host, it forwards the traffic based on its own routing table. This may send the traffic directly to the destination host, or it may be forwarded on to another router. This process of forwarding continues until the traffic reaches its final destination.



Gathering network information

Identifying Network Interfaces

The `ip link show` command will list all network interfaces available on your system:

```
[jmedinar@localhost ~]$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s25: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc state DOWN mode DEFAULT group default qlen 1000
    link/ether 3c:97:0e:b7:3f:e4 brd ff:ff:ff:ff:ff:ff
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DORMANT group default qlen 1000
    link/ether a4:4e:31:0d:90:50 brd ff:ff:ff:ff:ff:ff
```

To configure each network interface correctly, you need to know which one is connected to which network. In many cases, you will know the MAC address of the interface connected to each network, either because it is physically printed on the card or server, or because it is a virtual machine and you know how it is configured.

Displaying IP Addresses

Use the `ip addr show` command to view device and address information. A single network interface can have multiple IPv4 or IPv6 addresses.

```
[jmedinar@localhost ~]$ ip addr show wlp3s0
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state
UP group default qlen 1000
    link/ether a4:4e:31:0d:90:50 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.39/24 brd 192.168.1.255 scope global dynamic
noprefixroute wlp3s0
       valid_lft 68201sec preferred_lft 68201sec
    inet6 fe80::18ad:6abd:414f:60de/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

- An active interface state is UP.
- The link/ether line specifies the hardware (MAC) address of the device.
- The inet line shows an IPv4 address, its network prefix length, and scope.
- The inet6 line shows an IPv6 address, its network prefix length, and scope. This address is of global scope and is normally used.
- This inet6 line shows that the interface has an IPv6 address of link scope that can only be used for communication on the local Ethernet link.

Displaying Performance Statistics

The `ip -s link show` command may also be used to show statistics about network performance. Counters for each network interface can be used to identify the presence of network issues. The counters record statistics for things like the number of received (RX) and transmitted (TX) packets, packet errors, and packets that were dropped.

```
[jmedinar@localhost ~]$ ip -s link show wlp3s0
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state
UP mode DORMANT group default qlen 1000
    link/ether a4:4e:31:0d:90:50 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    1412387911 1265604  0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    63637927   506098   0       0       0       0
```

Displaying the Routing Table

Use the `ip route` command with the route option to show routing information.

```
[jmedinar@localhost ~]$ ip route
default via 192.168.1.1 dev wlp3s0 proto dhcp metric 600
192.168.1.0/24 dev wlp3s0 proto kernel scope link src 192.168.1.39
metric 600
```

This shows the `IPv4` routing table. All packets destined for the `192.168.1.0/24` network are sent directly to the destination through the device `wlp3s0`.

Add the -6 option to show the IPv6 routing table:

```
[jmedinar@localhost ~]$ ip -6 route
::1 dev lo proto kernel metric 256 pref medium
fe80::/64 dev wlp3s0 proto kernel metric 600 pref medium
```

Displaying status of network devices

The nmcli dev status command displays the status of all network devices:

```
[jmedinar@localhost ~]$ nmcli dev status
DEVICE      TYPE      STATE                   CONNECTION
wlp3s0      wifi      connected               loading-private
docker0     bridge    connected (externally)  docker0
virbr0      bridge    connected (externally)  virbr0
enp0s25     ethernet  unavailable             --
lo          loopback  unmanaged               --
virbr0-nic  tun       unmanaged               --
```

The nmcli con show command displays a list of all connections. To list only the active connections, add the --active option.

```
[jmedinar@localhost ~]$ nmcli con show
NAME                UUID                                  TYPE
DEVICE
loading-private     587180df-01f7-4f98-9eeb-43e95bb58479  wifi
wlp3s0
docker0             0ae44952-464c-4640-8d30-b21c89194678  bridge
docker0
virbr0              643392aa-3c8e-4b24-a47d-2dbd42d705d5  bridge
virbr0
CougarWiFi          707cb698-bb56-4663-8121-2537adaa501e  wifi
 --
loading...          a4404367-3c42-4363-930f-20c85c9d84b2  wifi
 --
SpectrumSetup-7D    8358b974-a2ca-42e2-9382-a8c85d6dcd85  wifi
 --
Wired connection 1  feb4d793-7e49-398f-abfc-1711b93fdff8  ethernet
 --
```

Network Configurations

The root user can make any necessary network configuration changes with `nmcli`.

However, regular users that are logged in on the local console can also make many network configuration changes to the system.

They have to log in at the system's keyboard to either a text-based virtual console or the graphical desktop environment to get this control. The logic behind this is that if someone is physically present at the computer's console, it's likely being used as a workstation or laptop and they may need to configure, activate, and deactivate wireless or wired network interfaces at will. By contrast, if the system is a server in the datacenter, generally the only users logging in locally to the machine itself should be administrators.

Regular users that log in using ssh do not have access to change network permissions without becoming root.

You can use the `nmcli gen permissions` command to see what your current permissions are.

```
[root@localhost ~]# nmcli gen permissions
PERMISSION
VALUE
org.freedesktop.NetworkManager.checkpoint-rollback
 yes
org.freedesktop.NetworkManager.enable-disable-connectivity-check
 yes
org.freedesktop.NetworkManager.enable-disable-network
yes
org.freedesktop.NetworkManager.enable-disable-statistics
 yes
org.freedesktop.NetworkManager.enable-disable-wifi
 yes
org.freedesktop.NetworkManager.enable-disable-wimax
yes
org.freedesktop.NetworkManager.enable-disable-wwan
 yes
org.freedesktop.NetworkManager.network-control
 yes
org.freedesktop.NetworkManager.reload
yes
org.freedesktop.NetworkManager.settings.modify.global-dns
yes
org.freedesktop.NetworkManager.settings.modify.hostname
yes
org.freedesktop.NetworkManager.settings.modify.own
 yes
org.freedesktop.NetworkManager.settings.modify.system
yes
org.freedesktop.NetworkManager.sleep-wake
yes
org.freedesktop.NetworkManager.wifi.scan
 yes
org.freedesktop.NetworkManager.wifi.share.open
 yes
org.freedesktop.NetworkManager.wifi.share.protected
yes
```

Network Manager

NetworkManager is a daemon that monitors and manages network settings. In addition to the daemon, there is a GNOME Notification Area applet providing network status information. Command-line and graphical tools talk to NetworkManager and save configuration files in the`/etc/sysconfig/network-scripts` directory.

```
[jmedinar@localhost ~]$ ls -l /etc/sysconfig/network-scripts/
-rw-r--r--. 1 root root 366 Oct 11 15:11 ifcfg-my-wireless-
network...
-rw-------. 1 root root  29 Oct 11 15:11 keys-my-wireless-network...
```

Adding a network connection

The `nmcli con add` command is used to add new network connections.

The following command adds a new connection named `eno2` for the interface eno2, which gets IPv4 networking information using DHCP and autoconnects on startup. It also gets IPv6 networking settings by listening for router advertisements on the local link. The name of the configuration file is based on the value of the con-name option, eno2, and is saved to the `/etc/sysconfig/network-scripts/ifcfg-eno2` file.

```
[root@localhost ~]# nmcli con add con-name eno2 type ethernet ifname
eno2
Connection 'eno2' (5d978db4-8937-4aa5-bb1e-f6b9b5218cdf)
successfully added.
```

Controlling network connections

The `nmcli con up name` command activates the connection name on the network interface it is bound to. Note that the command takes the name of a connection, not the name of the network interface.

So let's check the names first

```
[root@localhost ~]# nmcli con show
NAME                UUID                                  TYPE
DEVICE
docker0             0ae44952-464c-4640-8d30-b21c89194678  bridge
docker0
virbr0              643392aa-3c8e-4b24-a47d-2dbd42d705d5  bridge
virbr0
CougarWiFi          707cb698-bb56-4663-8121-2537adaa501e  wifi
 --
eno2                5d978db4-8937-4aa5-bb1e-f6b9b5218cdf  ethernet
 --
Wired connection 1  feb4d793-7e49-398f-abfc-1711b93fdff8  ethernet
 --
```

Now we can activate the connection

```
[root@localhost ~]# nmcli con up static-eno2
```

The `nmcli dev disconnect device` command disconnects the network interface device and brings it down.

```
[root@host ~]# nmcli dev disconnect eno2
```

Modifying network connection settings

NetworkManager connections have two kinds of settings. There are static connection properties, configured by the administrator and stored in the configuration files in `/etc/sysconfig/network-scripts/ifcfg-*`. There may also be active connection data, which the connection gets from a DHCP server and which are not stored persistently.

To list the current settings for a connection, run the `nmcli con show name` command, where name is the name of the connection.

- Settings in lowercase are static properties that the administrator can change.
- Settings in all caps are active settings in temporary use for this instance of the connection.

```
[root@localhost ~]# nmcli con show my-wireless-network
connection.id:                          my-wireless-network
connection.uuid:                        dasda-01f7-4f98-9eeb-
43e95bb58479
connection.stable-id:                   --
connection.type:                        802-11-wireless
connection.interface-name:              wlp3s0
connection.autoconnect:                 yes
connection.autoconnect-priority:        0
connection.autoconnect-retries:         -1 (default)
connection.multi-connect:               0 (default)
connection.auth-retries:                -1
connection.timestamp:                   1609272097
... output truncated ...
```

The `nmcli con mod name` command is used to change the settings for a connection. These changes are also saved in the `/etc/sysconfig/network-scripts/ifcfg-name` file for the connection.

Deleting a network connection

The `nmcli con del name` command deletes the connection named name from the system, disconnecting it from the device and removing the file `/etc/sysconfig/network-scripts/ifcfg-name`.

```
[root@localhost ~]# nmcli con del eno2
Connection 'eno2' (5d978db4-8937-4aa5-bb1e-f6b9b5218cdf)
successfully deleted.
```

Manually configuring the network

It is also possible to configure the network by directly editing the connection configuration files.

Connection configuration files control the software interfaces for individual network devices. These files are usually named `/etc/sysconfig/network-scripts/ifcfg-name`, where name refers to the name of the device or connection that the configuration file controls. The following are standard variables found in the file used for static or dynamic IPv4 configuration.

STATIC	DYNAMIC	EITHER
BOOTPROTO=none	BOOTPROTO=dhcp	DEVICE=ens3
IPADDR0=172.25.250.10		NAME="static-ens3"
PREFIX0=24		ONBOOT=yes
GATEWAY0=172.25.250.254		UUID=f3e8(...)ad3e
DEFROUTE=yes		USERCTL=yes
DNS1=172.25.254.254		

In the static settings, variables for IP address, prefix, and gateway have a number at the end. This allows multiple sets of values to be assigned to the interface. The DNS variable also has a number used to specify the order of lookup when multiple servers are specified.

After modifying the configuration files, run `nmcli con reload` to make NetworkManager read the configuration changes. The interface still needs to be restarted for changes to take effect.

```
1 [root@localhost ~]# nmcli con reload
2 [root@localhost ~]# nmcli con down "static-ens3"
3 [root@localhost ~]# nmcli con up "static-ens3"
```

Changing the hostname

The hostname command displays or temporarily modifies the system's fully qualified host name.

```
1 [root@localhost ~]# hostname
2 localhost.localdomain
3 [root@localhost ~]# hostname temp.hostname
4 [root@localhost ~]# hostname
5 temp.hostname
```

A static host name may be specified in the `/etc/hostname` file.

```
[root@localhost ~]# cat /etc/hostname
localhost.localdomain
```

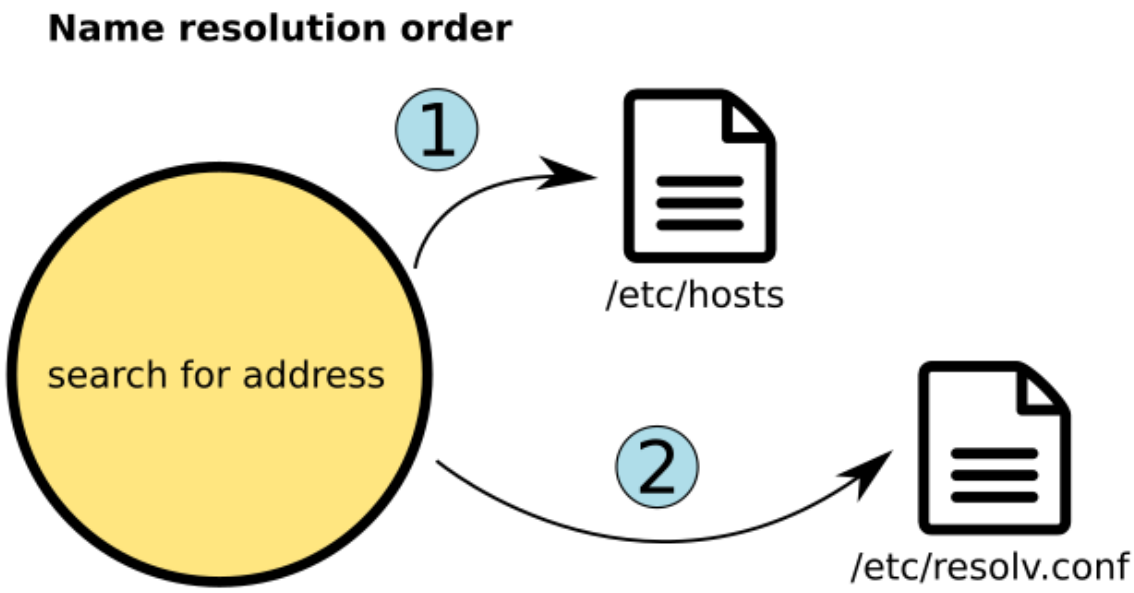
The `hostnamectl` command is used to modify this file and may be used to view the status of the system's fully qualified host name. If this file does not exist, the host name is set by a reverse DNS query once the interface has an IP address assigned.

```
[root@localhost ~]# hostnamectl
   Static hostname: localhost.localdomain
Transient hostname: temp.hostname
         Icon name: computer-laptop
           Chassis: laptop
        Machine ID: be54ea963f8b493dac8eff001550f1ed
           Boot ID: d694981fb2cb4f6caecf44425248576e
  Operating System: Fedora 33 (Workstation Edition)
       CPE OS Name: cpe:/o:fedoraproject:fedora:33
            Kernel: Linux 5.9.16-200.fc33.x86_64
      Architecture: x86-64
```

Changing back our hostname

```
[root@localhost ~]# hostnamectl set-hostname localhost.localdomain
[root@localhost ~]# hostnamectl
   Static hostname: localhost.localdomain
Transient hostname: fedora
         Icon name: computer-laptop
           Chassis: laptop
        Machine ID: be54ea963f8b493dac8eff001550f1ed
           Boot ID: d694981fb2cb4f6caecf44425248576e
  Operating System: Fedora 33 (Workstation Edition)
       CPE OS Name: cpe:/o:fedoraproject:fedora:33
            Kernel: Linux 5.9.16-200.fc33.x86_64
      Architecture: x86-64
[root@localhost ~]# hostname
localhost.localdomain
```

Configuring name resolution



The stub resolver is used to convert host names to IP addresses or the reverse. It determines where to look based on the configuration of the `/etc/nsswitch.conf` file. By default, the contents of the `/etc/hosts` file are checked first.

```
[root@localhost ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localdomain6
```

If an entry is not found in the `/etc/hosts` file, by default the stub resolver tries to look up the hostname by using a DNS nameserver.

The `/etc/resolv.conf` file controls how this query is performed:

- search: a list of domain names to try with a short host name. Both this and domain should not be set in the same file; if they are, the last instance wins.
- nameserver: the IP address of a nameserver to query. Up to three nameserver directives may be given to provide backups if one is down.

```
[root@localhost ~]# cat /etc/resolv.conf
# Generated by NetworkManager
domain example.com
search example.com
nameserver 172.25.254.254
```

NetworkManager updates the `/etc/resolv.conf` file using DNS settings in the connection configuration files.

```
[root@localhost ~]# nmcli con mod ID ipv4.dns IP
[root@localhost ~]# nmcli con down ID
[root@localhost ~]# nmcli con up ID
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-ID
...output omitted...
DNS1=8.8.8.8
...output omitted...
```

The default behavior of `nmcli con mod ID ipv4.dns IP`is to replace any previous DNS settings with the new IP list provided. A + or - symbol in front of the ipv4.dns argument adds or removes an individual entry.

```
[root@localhost ~]# nmcli con mod ID +ipv4.dns IP
```

Testing DNS Name Resolution

The`host HOSTNAME` command can be used to test DNS server connectivity.

```
[root@localhost ~]# host classroom.example.com
classroom.example.com has address 172.25.254.254
[root@localhost ~]# host 172.25.254.254
254.254.25.172.in-addr.arpa domain name pointer
classroom.example.com.
```

Monitoring & Troubleshooting

Checking connectivity between hosts

The ping command is used to test connectivity. The ping command sends a special network packet called an IMCP ECHO_REQUEST to a specified host. The command continues sending packets every second until `Ctrl+C` is pressed unless options are given to limit the number of packets sent.

```
[jmedinar@localhost ~]$ ping -c3 collin.edu
PING collin.edu (192.231.40.28) 56(84) bytes of data.
64 bytes from 192.231.40.28 (192.231.40.28): icmp_seq=1 ttl=113
time=16.8 ms
64 bytes from 192.231.40.28 (192.231.40.28): icmp_seq=2 ttl=113
time=16.4 ms
64 bytes from 192.231.40.28 (192.231.40.28): icmp_seq=3 ttl=113
time=16.8 ms

--- collin.edu ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 16.367/16.681/16.847/0.222 ms
```

The ping6 command is the IPv6 version of ping in Red Hat Enterprise Linux. It communicates over IPv6 and takes IPv6 addresses, but otherwise works like ping.

```
[user@host ~]$ ping6 2001:db8:0:1::1
PING 2001:db8:0:1::1(2001:db8:0:1::1) 56 data bytes
64 bytes from 2001:db8:0:1::1: icmp_seq=1 ttl=64 time=18.4 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=2 ttl=64 time=0.178 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=3 ttl=64 time=0.180 ms
^C
--- 2001:db8:0:1::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.178/6.272/18.458/8.616 ms
[user@host ~]$
```

When the remote host is not reachable we will get the following error

```
[jmedinar@localhost ~]$ ping 192.168.1.66
PING 192.168.1.66 (192.168.1.66) 56(84) bytes of data.
From 192.168.1.39 icmp_seq=1 Destination Host Unreachable

--- 192.168.1.66 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time
0ms

[jmedinar@localhost ~]$ ping6 2001:db8:0:1::1
ping6: connect: Network is unreachable
```

ping also helps you identify slow network responses.

Tracing routes

To trace the path that network traffic takes to reach a remote host through multiple routers, use either `traceroute` or `tracepath`. This can identify whether an issue is with one of your routers or an intermediate one. Both commands use UDP packets to trace a path by default; however, many networks block UDP and ICMP traffic. The traceroute command has options to trace the path with UDP (default), ICMP (-I), or TCP (-T) packets. Typically, however, the traceroute command is not installed by default.

```
[jmedinar@localhost ~]$ tracepath collin.edu
```

```
 1?: [LOCALHOST]                      pmtu 1500
 1:  _gateway                                              3.653ms
 1:  _gateway                                              2.165ms
 2:  142.254.130.213                                      10.658ms
 3:  no reply
 4:  agg24.plantxmp01r.texas.rr.com                       17.739ms
 5:  agg27.crtntxjt01r.texas.rr.com                       17.816ms
 6:  agg21.dllatxl301r.texas.rr.com                       11.855ms
 7:  66.109.1.216                                         38.482ms
 8:  209-18-43-77.dfw10.tbone.rr.com                      20.487ms
asymm  9
 9:  no reply
10:  ae2.cs1.dfw2.us.zip.zayo.com                         14.790ms
asymm 15
11:  ae1.mcs1.dfw2.us.eth.zayo.com                        14.259ms
12:  ae5.mpr1.dfw5.us.zip.zayo.com                        12.609ms
13:  208.184.23.101                                       14.014ms
...output omitted...
```

Each line in the output of `tracepath` represents a router or hop that the packet passes through between the source and the final destination. Additional information is provided as available, including the round trip timing (RTT) and any changes in the maximum transmission unit (MTU) size. The asymm indication means traffic reached that router and returned from that router using different (asymmetric) routes. The routers shown are the ones used for outbound traffic, not the return traffic.

The tracepath6 and traceroute -6 commands are the equivalent to tracepath and traceroute for IPv6.

Similar result with `traceroute`

```
[jmedinar@localhost ~]$ traceroute collin.edu
traceroute to collin.edu (192.231.40.28), 30 hops max, 60 byte
packets
 1  _gateway (192.168.1.1)  2.561 ms  2.625 ms  2.594 ms
 2  142.254.130.213 (142.254.130.213)  15.507 ms  15.569 ms  15.556
ms
 3  tge0-0-8.frsdtx1701h.texas.rr.com (70.125.219.93)  25.760 ms
 25.790 ms  25.770 ms
 4  agg24.plantxmp01r.texas.rr.com (24.175.49.227)  23.226 ms
 23.304 ms  23.264 ms
 5  agg27.crtntxjt01r.texas.rr.com (24.175.36.177)  18.265 ms
 23.235 ms  18.114 ms
 6  agg21.dllatxl301r.texas.rr.com (24.175.49.0)  18.195 ms  13.736
ms  13.631 ms
 7  bu-ether24.dllstx976iw-bcr00.tbone.rr.com (66.109.6.52)  18.560
ms  13.377 ms  13.338 ms
 8  66.109.5.121 (66.109.5.121)  13.108 ms  13.885 ms  11.967 ms
 9  * * *
10  ae2.cs1.dfw2.us.zip.zayo.com (64.125.26.202)  17.992 ms  18.262
ms  18.282 ms
11  ae1.mcs1.dfw2.us.eth.zayo.com (64.125.28.125)  17.859 ms
 17.852 ms  17.810 ms
```

```
14  12  ae5.mpr1.dfw5.us.zip.zayo.com (64.125.28.86)  17.738 ms  17.547
    ms  12.127 ms
15  13  208.184.23.101 (208.184.23.101)  12.100 ms  17.540 ms  17.151
    ms
```

In the output, we can see that connecting from our system to `collin.edu` requires traversing at least 12 routers. For routers that provided identifying information, we see their hostnames, IP addresses, and performance data, which includes three samples of round-trip time from the local system to the router. For routers that do not provide identifying information (because of router configuration, network congestion, firewalls, etc.), we see asterisks.

Checking ports and services

TCP services use sockets as end points for communication and are made up of an IP address, protocol, and port number. Services typically listen on standard ports while clients use a random available port. Well-known names for standard ports are listed in the `/etc/services` file.

The `ss` command is used to display socket statistics. The `ss` command is meant to replace the older tool `netstat`, part of the `net-tools` package, which may be more familiar to some system administrators but which is not always installed.

```
1  [jmedinar@localhost ~]$ ss -ta
2  State      Recv-Q      Send-Q      Local Address:Port      Peer
   Address:Port      Process
3  LISTEN     0           4096      0.0.0.0:hostmon
    0.0.0.0:*
4  LISTEN     0           32        192.168.122.1:domain
   0.0.0.0:*
5  LISTEN     0           4096      127.0.0.53%lo:domain
   0.0.0.0:*
6  LISTEN     0           128       127.0.0.1:ipp
    0.0.0.0:*
7  ESTAB      0           0         192.168.1.39:40090
   192.168.1.9:8008
8  ESTAB      0           0         192.168.1.39:50954
   192.168.1.9:8009
9  ESTAB      0           0         192.168.1.39:56884
   216.58.194.110:https
10 ESTAB      0           0         192.168.1.39:52702
   192.168.1.8:8009
11 ESTAB      0           0         192.168.1.39:33460
   31.13.93.53:https
12 ESTAB      0           0         192.168.1.39:54006
   64.233.180.188:hpvroom
13 ESTAB      0           0         192.168.1.39:52294
   23.4.41.235:https
14 ESTAB      0           0         192.168.1.39:56038
   34.216.200.139:https
15 ESTAB      0           0         192.168.1.39:38356
   13.249.29.43:https
```

```
16  LISTEN     0           4096       [::]:hostmon            [::]:*
17  LISTEN     0           128        [::1]:ipp               [::]:*
```

Options for ss and netstat

OPTION	DESCRIPTION
-n	Show numbers instead of names for interfaces and ports.
-t	Show TCP sockets.
-u	Show UDP sockets.
-l	Show only listening sockets.
-a	Show all (listening and established) sockets.
-p	Show the process using the sockets.
-A inet	Display active connections (but not listening sockets) for the inet address family. That is, ignore local UNIX domain sockets. For ss, both IPv4 and IPv6 connections are displayed. For netstat, only IPv4 connections are displayed. (netstat -A inet6 displays IPv6 connections, and netstat -46 displays IPv4 and IPv6 at the same time.)

netstat

The netstat program is used to examine various network settings and statistics. Through the use of its many options, we can look at a variety of features in our network.

```
1  [jmedinar@localhost ~]$ netstat -ie
2  Kernel Interface table
3  lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
4          inet 127.0.0.1  netmask 255.0.0.0
5          inet6 ::1  prefixlen 128  scopeid 0x10<host>
6          loop  txqueuelen 1000  (Local Loopback)
7          RX packets 247  bytes 24817 (24.2 KiB)
8          RX errors 0  dropped 0  overruns 0  frame 0
9          TX packets 247  bytes 24817 (24.2 KiB)
10         TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
11
12 wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
13         inet 192.168.1.39  netmask 255.255.255.0  broadcast
   192.168.1.255
14         inet6 fe80::18ad:6abd:414f:60de  prefixlen 64  scopeid
   0x20<link>
15         ether a4:4e:31:0d:90:50  txqueuelen 1000  (Ethernet)
16         RX packets 1291758  bytes 1424192191 (1.3 GiB)
17         RX errors 0  dropped 0  overruns 0  frame 0
18         TX packets 517621  bytes 65418705 (62.3 MiB)
```

```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The netstat command has hundreds of options but is more and more being replaced by the ss command!

```
[jmedinar@localhost ~]$ netstat --help
usage: netstat [-vWeenNcCF] [<Af>] -r         netstat {-V|--version|-h|--help}
       netstat [-vWnNcaeol] [<Socket> ...]
       netstat { [-vWeenNac] -I[<Iface>] | [-veenNac] -i | [-cnNe] -M | -s [-6tuw] } [delay]

        -r, --route              display routing table
        -I, --interfaces=<Iface> display interface table for <Iface>
        -i, --interfaces         display interface table
        -g, --groups             display multicast group memberships
        -s, --statistics         display networking statistics (like SNMP)
        -M, --masquerade         display masqueraded connections

        -v, --verbose            be verbose
        -W, --wide               don't truncate IP addresses
        -n, --numeric            don't resolve names
        --numeric-hosts          don't resolve host names
        --numeric-ports          don't resolve port names
        --numeric-users          don't resolve user names
        -N, --symbolic           resolve hardware names
        -e, --extend             display other/more information
        -p, --programs           display PID/Program name for sockets
        -o, --timers             display timers
        -c, --continuous         continuous listing

        -l, --listening          display listening server sockets
        -a, --all                display all sockets (default: connected)
        -F, --fib                display Forwarding Information Base (default)
        -C, --cache              display routing cache instead of FIB
        -Z, --context            display SELinux security context for sockets

  <Socket>={-t|--tcp} {-u|--udp} {-U|--udplite} {-S|--sctp} {-w|--raw}
            {-x|--unix} --ax25 --ipx --netrom
  <AF>=Use '-6|-4' or '-A <af>' or '--<af>'; default: inet
```

Secure Communications

Linux can be administered remotely over the network and of course to do so it has to be over a secure channel by using the SSH protocol implemented by the OpenSSH tool.

What is OpenSSH?

OpenSSH implements the Secure Shell or SSH protocol in Linux systems. The SSH protocol enables systems to communicate in an encrypted and secure fashion over an insecure network. You can use the `ssh` command to create a secure connection to a remote system, authenticate as a specific user, and get an interactive shell session on the remote system as that user. You may also use the `ssh` command to run an individual command on the remote system without running an interactive shell.

SSH is a client-server service so it must be running in both ends of the connection on port 22.

We can connect to a remote server `remotehost` using the same user name as the current local user.

```
[user01@localhost ~]$ ssh remotehost
user01@remotehost's password: ******
...output omitted...
[user01@remotehost ~]$
```

You can the exit command to log out of the remote system.

```
[user01@remotehost ~]$ exit
logout
Connection to remotehost closed.
[user01@localhost ~]$
```

We can also connect the remote server `remotehost` using a different username.

```
[user01@localhost ~]$ ssh user02@remotehost
user02@remotehost's password: ******
...output omitted...
[user02@remotehost ~]$
```

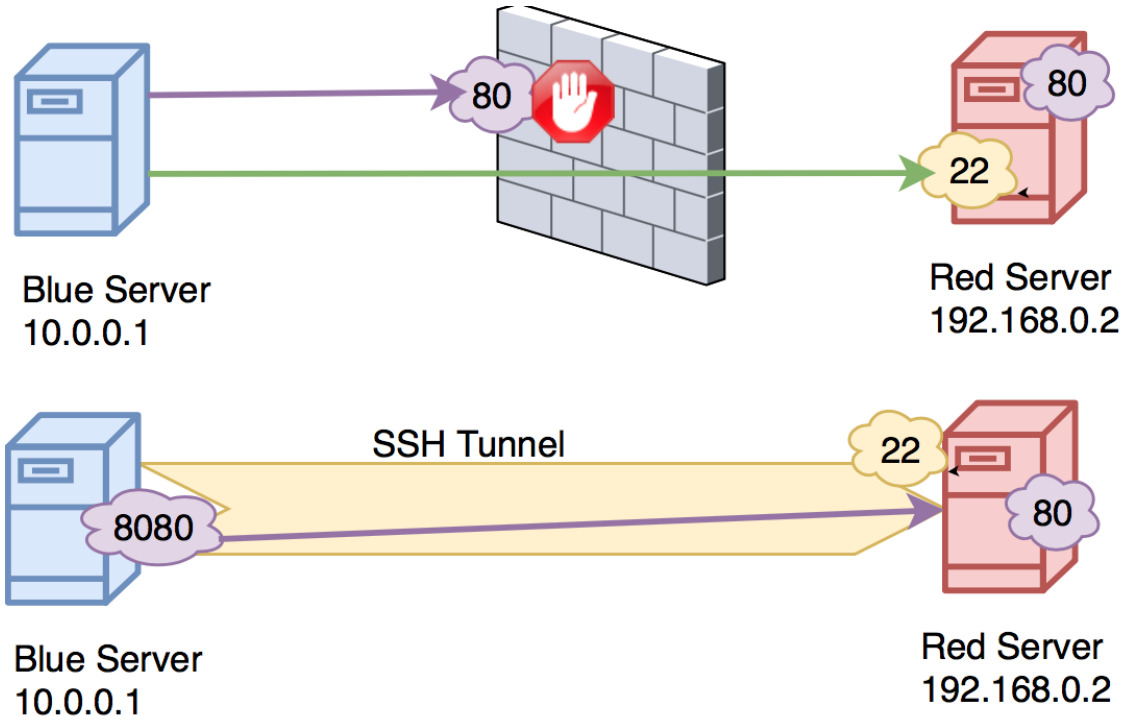
We can execute a command on the remote host as any user user without accessing the remote interactive shell.

```
[user01@localhost ~]$ ssh user02@remotehost hostname
user02@remotehost's password: ******
remotehost.lab.example.com
[user01@localhost ~]$
```

Notice that the preceding command displayed the output in the local system's terminal.

SSH client for Windows?

The most popular one is probably `PuTTY` by Simon Tatham and his team. The `PuTTY` program displays a terminal window and allows a Windows user to open an SSH session on a remote host. `PuTTY` is available at http://www.chiark.greenend.org.uk/~sgtatham/putty/



🐿 How SSH Works

SSH Keys

Generating a key pair provides you with two long string of characters: a public and a private key. You can place the public key on any server, and then unlock it by connecting to it with a client that already has the private key. When the two matches, the system unlocks without the need for a password. You can increase security even more by protecting the private key with a passphrase.

Step One—Create the RSA Key Pair

```
[jmedinar@localhost ~]$ ssh-keygen -t rsa
```

Step Two—Store the Keys and Passphrase

Once you have entered the Gen Key command, you will get a few more questions:

```
[jmedinar@localhost ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jmedinar/.ssh/id_rsa):
Created directory '/home/jmedinar/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jmedinar/.ssh/id_rsa.
Your public key has been saved in /home/jmedinar/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:uJkNEBO15FkwOcootMwifTs5EnHb3FLVEEdvZ4lz7Ok jmedinar@localhost.localdomain
The key's randomart image is:
+---[RSA 3072]----+
|    +o=o..+=o    |
| .. .=o=.  ...o .|
|+..=.=++     oo+o|
|+++ =.+..    .+o.|
|o. o oo.S      o |
|  . =  *      .  |
|   . o+ .      E |
|                 |
|                 |
+----[SHA256]-----+
```

You can select where to save the keys, usually, the default value is the best place so just hit enter.

You can set a password for your keys. This is the same password you will use to access this machine where you are generating the keys. You can also leave it empty!

The public key is now located in `your home directory under .ssh/id_rsa.pub`. The private key (identification) is now located in the same location as `.ssh/id_rsa`.

```
[jmedinar@localhost ~]$ tree .ssh/
.ssh/
├── id_rsa
└── id_rsa.pub
```

Log in with an SSH private key

Now that you have an SSH key pair, you're ready to configure your systems so they can SSH, SCP or SFTP to your server using your private key. You have to provide a copy of your **PUBLIC KEY ONLY!!!** to the remote host where you want to connect! **NEVER THE PRIVATE KEY!!!.**

```
[jmedinar@localhost ~]$ cd .ssh/
[jmedinar@localhost .ssh]$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDIlPUoyOxvkei5WkEpBiGSTx5drQOyKRnyoml6
h1tQW07c3I52WwmIoOwr9z7Hr3coRGSVAJGhk1YyjfIIKdFyNXoyZkN7JSKUf6w213zi0hQNDnBA
w6xIpo42Wz0XtXVsOFYYULtxgXFxe0B6Uks32d8fRHtB12vgk6JdiFSy9ZzGjPj2yI0dLgR6Aqsw
Y6MegkUpJxWsMMjNPS7c= jmedinar@localhost.localdomain
[jmedinar@localhost .ssh]$
```

The content of this file is a strange long string that represents your encrypted key!. You have to copy this content into the ~/.ssh/authorized_keys file in the remote system. If the file doesn't exist you will have to create it.

```
$(umask 077 && touch ~/.ssh/authorized_keys)
[jmedinar@localhost ~]$ chmod 700 ~/.ssh
[jmedinar@localhost ~]$ chmod 600 ~/.ssh/authorized_keys
```

Next, edit the file `.ssh/authorized_keys` using your preferred editor. Copy and paste your `id_rsa.pub` file into the file.

You can now SSH or SFTP into your server using your private key. From the command line, you can use:

```
[jmedinar@localhost ~]$ ssh SYSUSER@x.x.x.x
```

If you didn't create your key in the default location, you'll need to specify the location:

```
[jmedinar@localhost ~]$ ssh -i ~/.ssh/custom_key_name
SYSUSER@x.x.x.x
```

If you're using a Windows SSH client, such as PuTTy, look in the configuration instructions

scp [Secure Copy]

The OpenSSH package also includes two programs that can make use of an SSH-encrypted tunnel to copy files across the network. The first, scp (secure copy) is used much like the familiar cp program to copy files. The most notable difference is that the source or destination pathnames may be preceded with the name of a remote host, followed by a colon character.

The format is ALWAYS!!!

```
[jmedinar@localhost ~]$ scp source  target
```

To copy a file from our local system to our remote system we can do:

```
[jmedinar@localhost ~]$ scp my_local_file.txt
root@remote_server:/root
```

and to copy from remote to local

```
[jmedinar@localhost ~]$ scp
root@remote_server:/root/my_remote_file.txt /tmp
```

sftp [Secure FTP]

Ia a secure implementation for the ftp (File Transfer Protocol) program but it uses an SSH encrypted tunnel. sftp has an important advantage over conventional ftp in that it does not require an FTP server to be running on the remote host. It only requires the SSH server. This means that any remote machine that can connect with the SSH client can also be used as an FTP-like server.

```
[jmedinar@localhost ~]$ sftp remote_server
```

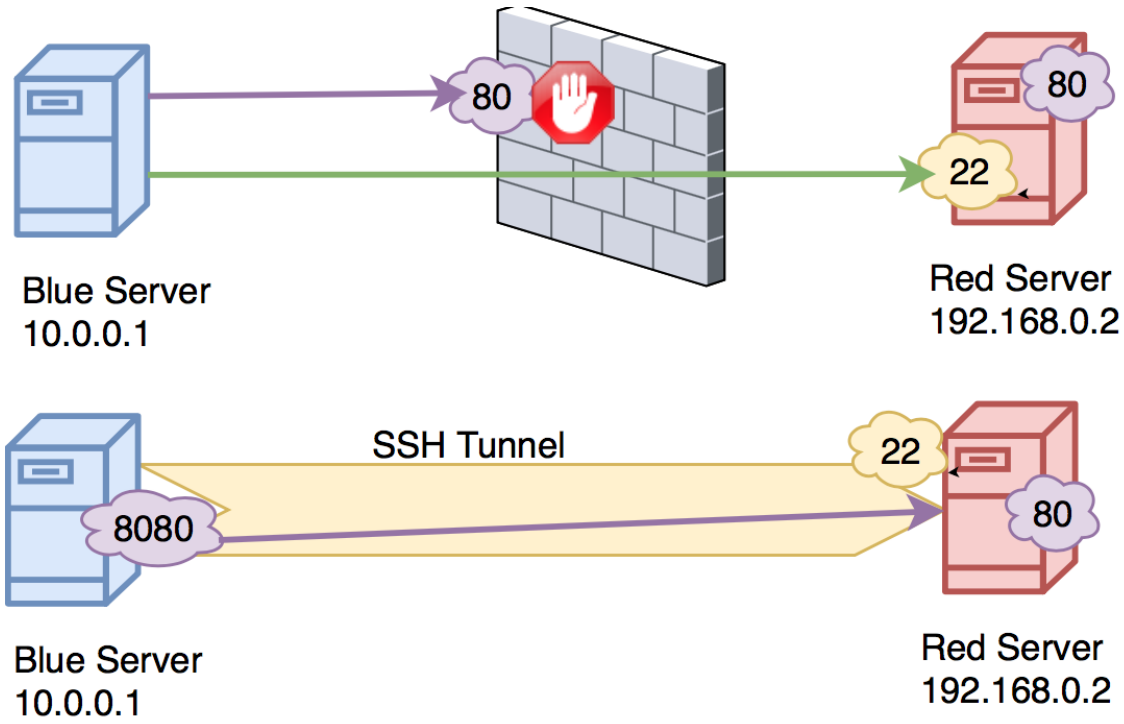
The SFTP protocol is supported by many of the graphical file managers found in Linux distributions. Like Filezilla.

Using FileZilla Secure FTP or SFTP Connection

http://www.chiark.greenend.org.uk/~sgtatham/putty/)

SSH Tunneling

Part of what happens when you establish a connection with a remote host via SSH is that an encrypted tunnel is created between the local and remote systems. Normally, this tunnel is used to allow commands typed at the local system to be transmitted safely to the remote system, and for the results to be transmitted safely back. In addition to this basic function, the SSH protocol allows most types of network traffic to be sent through the encrypted tunnel, creating a sort of VPN (Virtual Private Network) between the local and remote systems.



SSH Tunneling